Автономная некоммерческая организация
дополнительного профессионального образования
«Академия Информационных Технологий»

ОГРН: 1230600003457, ИНН/КПП: 0600010064/060001001

Российская Федерация, 386001, Республика Ингушетия, городской округ Магас, город Магас, улица Н.С. Хрущева, дом 10

УТВЕРЖДЕНО

Приказом № 1-ОБР от «7» апреля 2025г.

ДОПОЛНИТЕЛЬНАЯ ОБЩЕОБРАЗОВАТЕЛЬНАЯ ОБЩЕРАЗВИВАЮЩАЯ ПРОГРАММА «УПРАВЛЕНИЕ ЦИФРОВЫМИ ПРОДУКТАМИ И СИСТЕМАМИ»

Магас 2025

1.Пояснительная записка

Программа «Управление цифровыми продуктами и системами» нацелена на формирование у слушателей практических компетенций в области технического управления IT-продуктами — от понимания работы инфраструктуры и разработки до стратегического планирования бизнеса и управления командой.

Программа построена по модульному принципу. Часть модулей программы является обязательной для изучения всеми слушателями, часть модулей для изучения выбирается слушателем.

Вариативность наполнения программы позволяет слушателю уделить в процессе обучения максимальное внимание тем вопросам, которые наиболее важны для него и для его профессиональной деятельности.

При выборе траектории обучения по направлению «Программирование и управление данными» слушатель обязан освоить модули 1-5. При выборе траектории обучения по направлению «Технологическое предпринимательство» слушатель обязан освоить модули 1, 6-9. Модуль 1 является обязательным для освоения всеми слушателями. Слушатель, выбравший направление обучения, должен освоить модули направления и пройти итоговую аттестацию. При этом слушатель имеет право проходить обучение по двум направлениям программы или осваивать отдельные модули вне рамок выбранного направления.

При изучении программы применяются дистанционные технологии, что позволяет слушателю обучаться без отрыва от производственной деятельности, в удобное для себя время. Дистанционное обучение осуществляется с использованием интернет-ресурсов.

Уровень освоения программы: продвинутый

Направленность (профиль) программы – техническая

Форма обучения: очно-заочная с применением ЭО и ДОТ. Программа реализуется исключительно с использованием дистанционных образовательных технологий

Платформа для обучения: GetCourse.

Технические требования для слушателя:

ПК/ноутбук с доступом в Интернет

Веб-браузер (Google Chrome, Яндекс.Браузер).

Программы для просмотра PDF-файлов и работы с офисными документами.

Условия обучения:

Доступ к учебным материалам, заданиям и тестам предоставляется в личном кабинете на платформе GetCourse 24/7.

1.1. Актуальность Программы

Актуальность программы связана с ростом спроса на специалистов и предпринимателей в сфере информационных технологий, обладающих фундаментальными знаниями об устройстве цифровых продуктов и принципах их разработки. Уникальность — в практико-ориентированном формате: слушатели последовательно осваивают операционную систему Linux, систему контроля версий Git, программирование на языке C, структурное и объектно-ориентированное программирование, алгоритмы, базы данных SQL, DevOps, прикладную разработку на выбор (Python, Java, Go, C#, JS), а также проходят интенсивный курс по запуску и масштабированию технологического бизнеса.

1.2. Категории обучающихся

Взрослые (от 18 лет). Программа рассчитана на начинающих разработчиков, студентов технических специальностей, системных администраторов, специалистов технической поддержки, DevOps-инженеров начального уровня, тестировщиков, аналитиков, IT-менеджеров, владельцев бизнеса, а также всех, кто планирует карьерный переход в IT-сферу.

1.3. Цель и задачи программы

Цель Программы – формирование у слушателей практических компетенций в области технического управления IT-продуктами и системами, включая как инженерную подготовку (работа с инфраструктурой, алгоритмами и языками программирования), так и предпринимательские навыки (запуск и масштабирование технологического бизнеса), в зависимости от выбранной образовательной траектории.

Реализация поставленной цели предусматривает решение ряда задач.

Задачи Программы

Трек 1. Программирование и управление данными

Освоить базовую IT-инфраструктуру: работу в Linux (файловая система, процессы, сеть, bash-скрипты), систему контроля версий Git (репозитории, ветки, GitHub) и основы программирования на C (компиляция GCC, отладка, работа с памятью через указатели и динамическое выделение).

Овладеть алгоритмизацией и структурами данных на C: обработка массивов, строк, пользовательских структур, файловый ввод/вывод, реализация базовых алгоритмов (сортировка, поиск, работа с графами).

Изучить объектно-ориентированное программирование на C++ с реализацией шаблонных контейнеров и освоить прикладную разработку на одном из языков по выбору (Python, Java, Go, C#, JavaScript, Swift, Kotlin), включая работу с базами данных, веб-фреймворками и JWT-авторизацией.

Освоить инфраструктурные решения: системное администрирование Linux, основы DevOps (Docker, CI/CD), язык SQL (создание запросов, JOIN, агрегация).

Трек 2. Технологическое предпринимательство

Освоить методологию Lean Startup и Custdev: понимание цикла Build → Measure → Learn, определение MVP, проверку гипотез через клиентские интервью, формулирование гипотез о проблеме, решении и цене.

Научиться оценивать рынок и масштабировать бизнес: рассчитывать TAM, SAM, SOM, анализировать конкурентов, выбирать модели масштабирования (экспансия, франчайзинг, органический рост).

Овладеть юнит-экономикой и продуктовым подходом: расчет CAC, LTV, ARPU, построение финансовой модели на 12 месяцев, приоритизация гипотез (ICE/RICE), проверка Product-Market Fit.

Изучить стратегические концепции и инструменты продукта: подрывные инновации, стратегию голубого океана, Jobs to Be Done (JTBD), Customer Journey Map (CJM), Kano-модель, Value Proposition Canvas.

Сформировать компетенции в правовых и финансовых аспектах: выбор ОПФ и налогового режима (ИП, ООО, УСН, НПД), защита интеллектуальной собственности, управление налоговыми рисками и договорная работа.

1.4. Сроки реализации

Срок реализации:36-38 недель

Трудоемкость: 248 часов

Навыки

Работа в командной строке Linux (навигация, управление файлами, права доступа, процессы);

Использование Git (репозитории, коммиты, ветки, работа с GitHub);

Написание, компиляция и отладка программ на C (включая указатели, динамическую память, структуры, файлы);

Написание bash-скриптов для автоматизации задач;

Проведение Custdev-интервью и проверка гипотез;

Расчет рынка (TAM, SAM, SOM) и юнит-экономики (CAC, LTV, ARPU);

Выбор организационно-правовой формы и налогового режима.

Знания

Базовые команды Linux, структура файловой системы, права доступа;

Принципы работы Git и процесс компиляции GCC;

Синтаксис C (типы данных, операторы, условия, циклы, функции, указатели);

Методологии Lean Startup (Build → Measure → Learn, MVP, pivot) и Custdev;

Способы оценки рынка и анализа конкурентов;

Основы финансового моделирования и юнит-экономики.

Совершенствуемые компетенции

Работа с профессиональными инструментами разработки (Linux, Git, GCC);

Понимание низкоуровневого управления памятью и данными;

Проверка бизнес-гипотез через эксперименты и интервью с клиентами;

Финансовое прогнозирование и стратегическое мышление;

Клиентоцентричный подход к разработке продукта.

1.6.Календарный учебный график

Срок обучения:

Направление «Программирование и управление данными» - 38 недель

Режим занятий: до 7 часов в неделю.

Направления «Технологическое предпринимательство» - 36 недель

Режим занятий: до 7 часов в неделю.

Программа реализуется исключительно с применением ЭО и ДОТ и календарный учебный график формируется по мере комплектования учебных групп.

Обучение по программе осуществляется поэтапно (дискретно), посредством освоения отдельных учебных модулей.

№ п/п	Наименование модулей	Количество недель	Количество часов
1.	Основы IT-инфраструктуры (обязательный для всех слушателей)	3	82
2.	Алгоритмизация и структурное программирование	14	48
3.	Объектно-ориентированное программирование	10	40
4.	Современная прикладная разработка	4	42
5.	Инфраструктурные решения и управление данными	6	32
6.	Старт бизнеса	10	56
7.	Рост бизнеса	12	44
8.	Продуктовый подход в бизнесе	8	44

9.	Азбука стартапа	2	18
10.	Итоговая аттестация	1	4

Программа реализуется исключительно с применением ЭО и ДОТ и календарный учебный график формируется по форме календарного рейтинг-плана курса (маршрута обучения), размещаемого в системе дистанционного обучения GetCourse.

2.Содержание программы

2.1. Учебный план

№п/п	Наименование разделов/дисциплин/модулей	Общая трудоемкость	Всего аудиторных часов	в т.ч		Самостоятельная работа	Промежуточная/Итоговая аттестация	Форма промежуточной аттестации/текущего контроля (при наличии)
				Лекции/теория	Практические занятия			
1	Основы IT-инфраструктуры (обязательный для всех слушателей)	82	68	46	22	12	2	Экзамен
Модули направления «Программирование и управление данными» (обязательные для данного направления)		162	107	46	61	47	8	
2	Алгоритмизация и структурное программирование	48	32	14	18	14	2	Экзамен
3	Объектно-ориентированное программирование	40	26	11	15	12	2	Экзамен
4	Современная прикладная разработка	42	28	12	16	12	2	Экзамен
5	Инфраструктурные решения и управление данными	32	21	9	12	9	2	Экзамен
Модули направления «Технологическое предпринимательство» (обязательные для данного направления)		162	109	49	60	45	8	
6	Старт бизнеса	56	36	16	20	18	2	Экзамен
7	Рост бизнеса	44	30	14	16	12	2	Экзамен
8	Продуктовый подход в бизнесе	44	30	14	16	12	2	Экзамен
9	Азбука стартапа	18	13	5	8	3	2	Экзамен

10	Итоговая аттестация	4					4	
10.1	Экзамен	4					4	Экзамен
	Итого часов							
	Направление «Программирование и управление данными»	248	175	92	83	59	14	
	Направления «Технологическое предпринимательство»	248	177	95	82	57	14	

2.2. Учебно-тематический план/Рабочие программы модулей

Наименование компонентов программы	Содержание учебного материала и формы организации деятельности слушателей	Всего (час.)
Модуль 1. Основы IT-инфраструктуры		82
Тема 1.1. *Введение в Linux.*	*Содержание теоретических занятий* 1. Знакомство с ОС Linux. Дистрибутивы. Интерфейс Ubuntu Дистрибутивы: что это такое, какие бывают. Ubuntu и MINT. Интерфейсы для дистрибутивов. Что такое эмулятор. Знакомство с интерфейсом Ubuntu. Как не потеряться в системе. Как подключиться к интернету, открыть проводник, настройки и т.д. Где какие кнопки расположены. Как открыть приложения. 2.Работа с файловой системой Понимание структуры файловой системы Linux. Работа с правами доступа к файлам и директориям. Поиск файлов и директорий. Куда сохраняются файлы, где устанавливаются программы. 3.Терминал / Bash. Знакомство с командной строкой. Основа работы с Linux. Что такое терминал и зачем он нужен. Bash - командный интерпретатор. Как открыть терминал. Базовые команды в терминале mkdir, ls, cd, sudo, apt install, upgrade, whoami, history, touch, nano, vim, micro, find path –name «name» Работа с файловой системой через терминал. Создание, копирование, перемещение и удаление файлов и директорий. 4. Текстовые редакторы. Скрипты Основы работы с Vim и Nano. Создание и редактирование текстовых файлов. Введение в написание простых Bash-скриптов. Переменные, условия, циклы. 5. Работа с процессами Запуск, остановка и управление процессами. Мониторинг ресурсов и процессов.	4
	Содержание практических занятий 1. Знакомство с ОС Linux. Дистрибутивы. Интерфейс Ubuntu Запуск эмулятора. Визуальное знакомство с ОС. Развитие базовых навыков работы с Linux 2. Магазин Ubuntu Работа с приложениями Ubuntu. Как открыть, найти и скачать приложения (пакеты). 3.Работа с файловой системой Создание директорий, операции с ними. Создание файлов. 4. Текстовые редакторы. Скрипты Запуск текстовых редакторов. Освоение простейших операций. 5. Работа с процессами Просмотр процессов. Управление процессами.	8
	СРС Файловая система Linux, работа с простейшими операциями над файлами.	2
Тема 1.2. *Системы контроля версий Git*	*Содержание теоретических занятий* 1. Установка Git Git и GitHub: в чем разница. Установка Git. 2. Знакомство с терминалом. Знакомство с терминалом. Директории: создание, перемещение, удаление. Схожесть с Bash.	10

	3. Основы Git. Git Cheat Sheet. Работа с командами (init, add, commit) Коммит. Создание репозитория. Создание файла. Сохранение файла в локальном репозитории. 4. GitHub Регистрация на GitHub. Создание удаленного репозитория. Команды (remote add, push) 5. Ветки Git Что такое ветка, для чего она используется, как её создать и перейти. Pull request 6. Графический интерфейс Git GUI - интерфейс Git.	
	Содержание практических занятий 1. Знакомство с терминалом. Работа в локальном репозитории. Разместить его в сети.	2
	СРС Обзорное знакомство слушателя с системой контроля версий Git и возможностями командной работы.	2
Тема 1.3. *Введение в язык С*	*Содержание теоретических занятий* 1. Структура программы. Основные компоненты программы на языке С. Синтаксис языка Си. Библиотека. Подключение библиотеки <stdio.h> для ввода/вывода данных. Код начинающего: Привет, мир! Среда разработки (блокнот). Компиляция языка в консоли. Онлайн-компилятор. 2. Функция вывода данных (printf). Общая форма записи функции printf(). Управляющие символы. Форматирование с помощью функции printf 3. Сборка и запуск программ (GCC). Процесс компиляции и выполнения программ на языке С. Компилятор GCC: установка, запуск, знакомство. Набор компиляторов. Исполняемые файлы. Команды (gcc main.c -o program) 4. Типы данных Основные типы данных и их использование. Int, float, double, char. 5. Переменные. Типы данных в языке Си. Переменные. Объявление переменных. 6. Функция ввода данных (scanf) Функция форматированного ввода данных с клавиатуры scanf() 7. Арифметические операторы Основные арифметические операторы языка Си. + оператор сложения - оператор вычитания * оператор умножения % оператор взятия остатка от деления / оператор деления 8. Операторы сравнения Равно. Не равно. Больше чем, меньше чем. Больше либо равно, меньше либо равно. 9. Логические операторы Логические операторы И/или 10. Условные операторы. Последовательность выполнения команд в программе, условные операторы (if else, switch). 11. Цикл For Использование цикла for 12. Цикл While Циклы while,do while, break-continue. 13. Введение в функции языка Си Что такое функция, ее типы. Тело функции.	18

	Вызов функции. Возвращаемое значение. Аргументы функции. Системные и пользовательские функции. Рекурсия.	
	Содержание практических занятий 1. Структура программы. Основные компоненты программы на языке С. Запуск онлайн-копилятора. Отработка первых строк кода. Знакомство с синтаксисом. 2. Функция вывода данных (printf). Тестирование. Практическая работа в онлайн-компиляторе.	8
	СРС Запуск онлайн-компилятора. Визуальное знакомство с синтаксисом языка. Выполнение практических заданий.	2
Тема 1.4. *Указатель и работа с памятью*	*Содержание теоретических занятий* 1. Понятие указателя. Определение и использование указателей. Память компьютера. Определение указателя. Использование указателей. Разыименование. Операция получения адреса. Работа с адресами. 2. Адресная арифметика Арифметические действия над указателями. Сравнение. Увеличение/ уменьшение. Сложение, вычитание, индексация. Отличие от арифметических выражений. 3. Массивы Массив: что такое и каких типов бывают. Одномерные/двумерные/многомерные Объявление, инициализация и использование массивов. 4. Алгоритмы с массивами Поиск, сортировка и другие алгоритмы. Сортировка пузырьком.	4
	Содержание практических занятий 1. Понятие указателя. Определение и использование указателей. Работа в компиляторе.	4
	СРС Обзорное знакомство с операциями, которые могут быть применены к массивам.	2
Тема 1.5. *Работа с текстом и пользовательскими структурами данных*	*Содержание теоретических занятий* 1. Символы. Работа с символьными данными. Представление символов в Си. Кодовые таблицы. Таблица ASCII. Управляющие символы 2. Строки. Обработка строковых данных. Строка. Строковая константа. Связь строк с указателями, с массивами. Оператор sizeof. 3. Структуры. Определение и использование пользовательских структур данных. Что такое структура. Поля структуры. Struct. Как создавать структуры, объявлять их. Как получать доступ к полям структуры. Определение и использование пользовательских структур данных.	6
	Содержание практических занятий	0
	СРС Представление символов в языке Си, кодовые таблицы, такие как ASCII.	2
Тема 1.6. *Работа с файлами*	*Содержание теоретических занятий* 1. Текстовые файлы Тип данных для работы с файлами. Тип File. Как открывать/закрывать файлы. Режимы работы с файлами. Операции с текстовыми файлами: ввод/вывод данных из файла. 2. Бинарные файлы. Работа с бинарными файлами.	4

	Что такое бинарные файлы. Чтение и запись бинарных файлов. Функции для работы	
	Содержание практических занятий	0
	СРС Знакомство с особенностями работы с файлами в языке Си.	2
Промежуточная аттестация	*Экзамен*	2
Модуль 2. Алгоритмизация и структурное программирование		48
Тема 2.1 Структурное программирование (на языке С)	*Содержание теоретических занятий* 1. Утилиты командной строки Bash для работы с текстом Стандартные потоки ввода/вывода. Аргументы командной строки и флаги утилит. Регулярные выражения (для grep). Построчная обработка текстовых файлов. Коды возврата и обработка ошибок 2. Библиотека строковых функций на языке Си Нуль-терминированные строки в Си. Указатели и работа с памятью. Стандартные функции string.h (назначение и принципы работы). Форматированный вывод (printf, sprintf). Форматированный ввод (scanf, sscanf). Функции с переменным числом параметров (stdarg.h) 3. Тип данных decimal для точных финансовых расчётов Проблемы точности двоичной арифметики для финансов. Десятичное представление чисел (BCD-подобное). Структура типа decimal: знак, масштаб, мантисса. Арифметика с фиксированной десятичной точкой. Обработка переполнения и недопустимых операций. 4. Библиотека для работы с матрицами Матрицы: размерность, индексация. Динамическое выделение памяти под матрицу. Операции сложения, вычитания, умножения матриц. Транспонирование матрицы. Вычисление определителя. Понятие обратной матрицы и условия её существования. Алгебраические дополнения 5. Консольная игра BrickGame (Тетрис) Конечные автоматы (состояния, переходы, события) Игровой цикл и управление временем. Игровое поле и фигуры (тетрамино). Обработка пользовательского ввода (консоль). Отрисовка в консоли. Система подсчёта очков и рекордов	10
	Содержание практических занятий 1. Утилиты командной строки Bash для работы с текстом Реализация утилиты cat. Реализация утилиты less. Реализация утилиты more. Реализация утилиты grep. Обработка нескольких файлов. Unit-тестирование 2. Библиотека строковых функций на языке Си Реализация всех функций библиотеки string.h. Реализация функции sprintf. Реализация функции sscanf. Полное покрытие unit-тестами 3. Тип данных decimal для точных финансовых расчётов Реализация структуры s21_decimal. Реализация арифметических операций (сложение, вычитание, умножение, деление). Реализация логических операций (and, or, xor, not). Реализация операций сравнения (равно, не равно, больше, меньше, больше или равно, меньше или равно). Обязательное покрытие unit-тестами 4. Библиотека для работы с матрицами Реализация структуры для матрицы. Создание и уничтожение матрицы. Изменение элементов матрицы. Сложение матриц. Вычитание матриц. Умножение матриц (включая умножение на число). Транспонирование. Вычисление определителя. Поиск обратной матрицы. Unit-тестирование всех операций 5. Консольная игра BrickGame (Тетрис) Разработка законченного прикладного приложения на С.	10

	Реализация игровой логики (тетрис-лайк). Реализация консольного пользовательского UI. Применение полезных алгоритмов и структур данных. Управление из консоли. Запись и хранение рекордов. Отсутствие ошибок и утечек памяти	
	СРС Изучение man-страниц утилит cat, less, more, grep и регулярных выражений. Разбор работы функций с переменным числом параметров (stdarg.h) и форматных строк printf/scanf. Изучение стандарта IEEE 754, BCD и спецификации типа decimal в других языках. Анализ алгоритмов вычисления определителя и обратной матрицы (Гаусс, Гаусс-Жордан). Изучение теории конечных автоматов и разработка диаграммы состояний для игры. Освоение консольного ввода-вывода (ncurses / ANSI escape-коды). Реализация дополнительных функций по выбору (округление decimal, решение СЛАУ, таблица рекордов и др.). Сравнительный анализ производительности и сложности алгоритмов (Big O, бенчмарки).	10
Тема 2.2 *Введение в алгоритмы (язык по выбору C, Python, C#, Java, JavaScript, Go, Swift, Kotlin)*	*Содержание теоретических занятий* 1. Поиск выхода из лабиринта Представление лабиринта в виде графа (клетки — вершины, проходы — рёбра). Алгоритмы поиска пути в графе (нерекуррентные): BFS (поиск в ширину), стековый DFS (итеративный), алгоритм Дейкстры (если с весами). Отличие рекуррентного и итеративного (нерекуррентного) подходов. Способы хранения лабиринта: матрица смежности, список рёбер, файл карты. Обработка тупиков и множественных путей 2. Построение маршрута на взвешенном графе (поиск кратчайшего пути) Взвешенные графы: вершины, рёбра, веса (расстояние, время, стоимость). Алгоритмы поиска кратчайшего пути: Дейкстра, A* (A-star), Флойда-Уоршелла (для всех пар вершин). Представление графа: матрица весов, список смежности. Форматы загрузки графа из файла (матрица, список рёбер). Восстановление маршрута по предкам.	4
	Содержание практических занятий 1. Поиск выхода из лабиринта Реализация загрузки карты лабиринта из файла. Реализация нерекуррентного алгоритма поиска выхода (BFS или итеративный DFS). Построение маршрута от старта до выхода. Визуализация пути (консольная или графическая). Unit-тестирование на разных лабиринтах 2. Построение маршрута на взвешенном графе (поиск кратчайшего пути) Реализация загрузки взвешенного графа из файла. Реализация алгоритма Дейкстры для поиска кратчайшего пути. Построение и вывод маршрута (последовательность вершин, общая длина). Обработка случаев: недостижимые вершины, пустой граф, отрицательные веса (если применимо). Unit-тестирование на разных графах	8
	СРС Изучение представления графов (матрица смежности, список рёбер, файловые форматы). Разбор алгоритмов BFS и DFS (итеративная реализация, отличие от рекурсивной). Изучение алгоритма Дейкстры и его применимости (отрицательные веса, производительность). Ознакомление с алгоритмом A* (эвристики, отличие от Дейкстры). Сравнение алгоритмов поиска пути по времени работы и памяти (Big O). Реализация загрузки лабиринта и взвешенного графа из разных форматов файлов. Визуализация	4

	найденного пути (консольная отрисовка или вывод маршрута). Написание unit-тестов для граничных случаев (нет пути, старт = финиш, одна вершина).	
Промежуточная аттестация	*Экзамен*	2
Модуль 3. Объектно-ориентированное программирование		40
Тема 3.1 Объектно-ориентированное программирование (на языке C++)	*Содержание теоретических занятий* 1. Объектно-ориентированная библиотека для работы с матрицами (перегрузка операторов, паттерн «Свойство») Основы ООП: классы, конструкторы, деструкторы, инкапсуляция. Перегрузка операторов (+, -, *, =, ==, +=, индексации [] и др.). Паттерн «Свойство» (property): геттеры и сеттеры с логикой, доступ через синтаксис поля. Управление динамической памятью в классе (правило трёх/пяти). Исключения и обработка ошибок 2. Реализация контейнеров стандартной библиотеки C++ (list, map, queue, set, stack, vector) + доп. array, multiset, matvector Шаблоны классов (template) в C++. Итераторы (категории, реализация begin/end). Структуры данных: двусвязный список, ассоциативные массивы (деревья/хэши), стек, очередь, динамический массив. Алгоритмическая сложность операций (O(1), O(n), O(log n)). Правило пяти для контейнеров. Дополнительно: счетчик (multiset), математический вектор (matvector) 3. Десктопная игра BrickGame (ООП, обратная совместимость) ООП-архитектура игры: классы (Game, Model, View, Controller). Паттерн MVC (Model-View-Controller). Обратная совместимость: поддержка старой игры и старого интерфейса + добавление новой игры и нового интерфейса. Десктопный UI (Qt, SFML, SDL или аналоги). Разделение логики и отображения 4. Каркасный 3D-вьювер (загрузка OBJ-файлов, отображение моделей, ОО-стиль) Формат файлов OBJ (вершины, грани, группы, материалы). Парсинг текстовых файлов (чтение вершин и полигонов). Аффинные преобразования в 3D: поворот, масштабирование, перенос. Матрицы преобразований (4×4, однородные координаты). Основы компьютерной графики: каркасная модель (wireframe), проекции (перспективная/ортографическая). Графические библиотеки (OpenGL, Qt OpenGL, или чистое окно).	11
	Содержание практических занятий 1. Объектно-ориентированная библиотека для работы с матрицами (перегрузка операторов, паттерн «Свойство») Реализация класса матрицы с ООП-подходом. Перегрузка арифметических и логических операторов. Реализация паттерна «Свойство» (например, для доступа к строкам/столбцам). Unit-тестирование 2. Реализация контейнеров стандартной библиотеки C++ (list, map, queue, set, stack, vector) + доп. array, multiset, matvector Реализация шаблонных классов vector, list, stack, queue, set, map. Реализация итераторов для каждого контейнера. Дополнительно: array, multiset (счётчик), matvector. Unit-тестирование (сравнение с эталонной std::). Проверка утечек памяти 3. Десктопная игра BrickGame (ООП, обратная совместимость) Разработка законченного десктопного приложения на C++. Реализация игровой логики (тетрис-лайк) в ОО-стиле. Реализация десктопного пользовательского интерфейса. Обеспечение обратной совместимости (старая игра + старый UI; новая игра + новый UI). Применение алгоритмов и структур данных. Сохранение рекордов. 4. Каркасный 3D-вьювер (загрузка OBJ-файлов, отображение	15

	моделей, OO-стиль) Реализация загрузчика .obj файлов. Построение каркасной 3D-модели на экране. Реализация управления (поворот, масштабирование, перенос мышью/клавишами). Реализация в объектно-ориентированном стиле. Отображение с возможностью переключения проекций	
	СРС Изучение правил трёх и пяти в C++ (управление ресурсами). Разбор перегрузки операторов (бинарные, унарные, индексные, приведения типов). Изучение шаблонов классов и итераторов (категории, реализация). Анализ сложности основных операций контейнеров (Big O для list, vector, map, set). Изучение паттерна MVC (Model-View-Controller) для игр и приложений. Разбор формата файлов OBJ (структура, парсинг вершин и граней). Изучение матриц аффинных преобразований в 3D (однородные координаты). Освоение основ графических библиотек (OpenGL / Qt) для отображения 3D-моделей	12
Промежуточная аттестация	*Экзамен*	2
Модуль 4. Современная прикладная разработка		42
Тема 4.1 *Прикладное программирование (язык по выбору Python, C#, Java, JavaScript, Go, Swift,* Kotlin)	*Содержание теоретических занятий* 1. Интенсив АР на языке C# Основные конструкции языка C#: типы данных, операторы, ветвления, циклы, массивы, строки, работа с консолью. Объектно-ориентированное программирование в C#: классы, объекты, наследование, полиморфизм, инкапсуляция, интерфейсы, абстрактные классы, статические члены. Разработка консольных игр: библиотека dotnet-curses, пошаговая логика, генерация карты, обработка ввода, отрисовка (на примере Rogue-like). Веб-разработка на ASP.NET Core: маршрутизация, контроллеры, представления (Razor), модели, обработка HTTP-запросов. Работа с базами данных: SQL, Entity Framework Core (миграции, LINQ, связи таблиц), CRUD-операции. Аутентификация и авторизация: cookie-аутентификация, роли, JWT (структура, генерация, валидация), защита API. 2. BrickGame v3.0 на C# Архитектура Web/Mobile приложений на C# (Blazor, ASP.NET Core API, MAUI, Xamarin — по выбору). Обратная совместимость программных компонентов: поддержка старой версии игры и старого интерфейса при добавлении новой игры и нового интерфейса. Паттерн MVC для игр с графическим интерфейсом. Взаимодействие клиента и сервера (API, SignalR). Сохранение рекордов и прогресса (локальное хранилище или база данных). 3. Интенсив АР на языке Java Основные конструкции языка Java: типы данных, операторы, ветвления, циклы, массивы, строки, работа с консолью. ООП в Java: классы, объекты, наследование, полиморфизм, инкапсуляция, интерфейсы, абстрактные классы. Применение процедурного и мультипарадигмального подхода. Написание кода с соблюдением функциональной парадигмы (лямбда-выражения, Stream API). Разработка консольных игр: библиотека JCurses для Java, пошаговая логика, генерация карты, обработка ввода, отрисовка (на примере Rogue-like). Веб-разработка на Spring: Spring Boot, контроллеры (RestController), маршрутизация, внедрение	12

	зависимостей (IoC/DI). Работа с базами данных: Spring Data JPA, Hibernate, миграции, CRUD-операции, связи между таблицами. Аутентификация и авторизация в Spring Security. JWT-авторизация: структура токена, генерация, валидация, настройка Spring Security для работы с JWT. Расширение возможностей веб-приложений (CORS, кэширование, обработка ошибок). Разработка мобильных приложений на Android: архитектура (Activity, Fragment), сетевые запросы (Retrofit), работа с JSON. Создание Android-клиента для сервера. Добавление JWT-авторизации в мобильное приложение. 4. BrickGame v3.0 на Java Архитектура Web/Mobile приложений на Java (Spring Boot для бэкенда, Android для мобильного клиента, Web-интерфейс по выбору). Обратная совместимость программных компонентов: поддержка старой версии игры и старого интерфейса при добавлении новой игры и нового интерфейса. Паттерн MVC для игр с графическим интерфейсом. Взаимодействие клиента и сервера (REST API, WebSocket при необходимости). Сохранение рекордов и прогресса (база данных, локальное хранилище на клиенте). 5. Интенсив АР на языке JavaScript Основы языка JavaScript: типы данных, переменные (var, let, const), операторы, ветвления, циклы, функции, массивы, объекты, работа с консолью. Применение ОП/процедурного/мультипарадигмального подхода в JavaScript: прототипное наследование, классы (ES6), инкапсуляция, полиморфизм. Написание кода с соблюдением функциональной парадигмы (функции высшего порядка, замыкания, чистота функций, иммутабельность, map, filter, reduce). Разработка консольных игр: библиотека blessed для JavaScript (NCurses-подобная), пошаговая логика, генерация карты, обработка ввода, отрисовка (на примере Rogue-like) в среде Node.js. Веб-разработка на Node.js и Nest.js: архитектура Nest.js (модули, контроллеры, провайдеры, сервисы), маршрутизация, внедрение зависимостей, Middleware, обработка HTTP-запросов. Создание клиентского приложения на Vue.js: компонентный подход, реактивность, маршрутизация (Vue Router), управление состоянием (Pinia/Vuex), взаимодействие с сервером через HTTP-клиент (Axios). Базы данных и авторизация: подключение базы данных к Nest.js (TypeORM/Prisma), работа с сущностями и отношениями, CRUD. Аутентификация и авторизация (JWT, Passport.js, guards), хеширование паролей. Расширение возможностей серверного приложения на Nest.js: обработка ошибок, валидация (class-validator), CORS, WebSockets (при необходимости), документация API (Swagger), тестирование (Jest). 6. BrickGame v3.0 на JavaScript Архитектура Web/Mobile приложений на JavaScript (бэкенд на Node.js/Nest.js, фронтенд на Vue.js/React/Angular или мобильный клиент на React Native/Ionic). Обратная совместимость программных компонентов: поддержка старой версии игры и старого интерфейса при добавлении новой игры и нового интерфейса. Паттерн MVC для игр с графическим интерфейсом (на клиенте). Взаимодействие клиента и сервера (REST API, WebSocket для реального времени при необходимости). Сохранение рекордов и прогресса (база данных на сервере, локальное хранилище на клиенте: localStorage/IndexedDB). 7. Интенсив АР на языке Kotlin Основы языка Kotlin: типы данных, переменные (val, var), операторы, ветвления, циклы, массивы, коллекции, строки, работа с консолью (readLine, print). Применение	

	ООП/процедурного/мультипарадигмального подхода в Kotlin: классы, объекты, наследование, полиморфизм, инкапсуляция, интерфейсы, абстрактные классы, data-классы, object-декларации, sealed-классы. Написание кода с соблюдением функциональной парадигмы (лямбда-выражения, функции высшего порядка, scope-функции (let, apply, with, run, also), работа с коллекциями (map, filter, reduce, fold)). Разработка консольных игр: использование cinterop для взаимодействия с библиотекой NCurses на C. Пошаговая логика, генерация карты, обработка ввода, отрисовка (на примере Rogue-like). Веб-разработка на Ktor: архитектура Ktor (модули, маршрутизация, плагины), обработка HTTP-запросов, Content Negotiation, Status Pages, CORS. Работа с базами данных: подключение БД к Ktor (Exposed — DSL/ORM, или JDBC), работа с сущностями, миграции (Flyway), CRUD-операции. Аутентификация и авторизация в Ktor: механизмы (Session, JWT, Basic), работа с паролями (хэширование). JWT-авторизация: структура токена, генерация, валидация, настройка JWT-плагина в Ktor, защита маршрутов. Разработка мобильных приложений на Android: архитектура (Activity, Fragment, ViewModel, LiveData/StateFlow), сетевые запросы (Retrofit + Moshi/Gson), работа с JSON, корутины (Coroutines) для асинхронности. Создание Android-клиента для сервера на Ktor. Добавление JWT-авторизации в Android-приложение (хранение токена, автоматическая подстановка в заголовки). 8. BrickGame v3.0 на Kotlin Архитектура Web/Mobile приложений на Kotlin (бэкенд на Ktor, мобильный клиент на Android, Web-интерфейс по выбору — например, на React/Vue с API на Ktor). Обратная совместимость программных компонентов: поддержка старой версии игры и старого интерфейса при добавлении новой игры и нового интерфейса. Паттерн MVC/MVVM для игр с графическим интерфейсом. Взаимодействие клиента и сервера (REST API, WebSocket при необходимости). Сохранение рекордов и прогресса (база данных на сервере, локальное хранилище на клиенте — SharedPreferences/Room). 9. Интенсив АР на языке Python Основы языка Python: типы данных (int, float, str, bool, None), переменные, операторы, ветвления (if-elif-else), циклы (for, while), списки, кортежи, словари, множества, строки, функции (def, lambda, аргументы, возвращаемые значения), работа с консолью (input, print). Применение ООП/процедурного/мультипарадигмального подхода в Python: классы, объекты, наследование, полиморфизм, инкапсуляция (соглашения о приватности), магические методы (__init__, __str__, __repr__, __eq__ и др.), декораторы (@staticmethod, @classmethod, @property). Написание кода с соблюдением функциональной парадигмы (функции высшего порядка, замыкания, декораторы, map, filter, reduce, lambda, list comprehensions). Разработка консольных игр: библиотека curses (встроенная в Python), управление экраном, цветом, клавиатурой (non-blocking input), пошаговая логика, генерация карты, отрисовка, игровой цикл (на примере Rogue-like). Веб-разработка на Flask: основы Flask (маршрутизация, представления, шаблоны Jinja2, запросы и ответы, сессии), обработка GET/POST, формы и валидация (WTForms при необходимости), работа с JSON (API). Работа с базами данных: подключение БД к Flask (SQLAlchemy ORM, Flask-SQLAlchemy, миграции Alembic), модели, связи между таблицами, CRUD-операции. Аутентификация и авторизация в	

	Flask: Flask-Login, хэширование паролей (bcrypt или werkzeug.security), управление пользователями и ролями. JWT-авторизация: структура JWT-токена, генерация и валидация, настройка JWT в Flask (Flask-JWT-Extended), защита маршрутов (декоратор @jwt_required), работа с refresh-токенами, хранение токенов на клиенте. Расширение возможностей веб-приложений на Flask: CORS (Flask-CORS), обработка ошибок (error handlers), кэширование (Flask-Caching), фоновая обработка задач (Celery + Redis), тестирование (pytest), документация API (Flask-Swagger/Flask-RESTX). 10. BrickGame v3.0 на Python Архитектура Web/Mobile приложений на Python (бэкенд на Flask/Django, фронтенд на HTML/CSS/JS или мобильный клиент на Kivy/Beeware/Flet). Обратная совместимость программных компонентов: поддержка старой версии игры и старого интерфейса при добавлении новой игры и нового интерфейса. Паттерн MVC для игр с графическим интерфейсом (на клиенте или на сервере с Web-интерфейсом). Взаимодействие клиента и сервера (REST API, WebSocket — Flask-SocketIO при необходимости). Сохранение рекордов и прогресса (база данных на сервере, локальное хранилище на клиенте: localStorage/IndexedDB для Web, SQLite для мобильного клиента). 11. Интенсив AP на языке Swift Основы языка Swift: типы данных (Int, Double, String, Bool), переменные (var) и константы (let), операторы, ветвления (if, guard, switch), циклы (for-in, while, repeat-while), массивы (Array), словари (Dictionary), множества (Set), строки и их методы, опционалы (Optional, unwrapping: if let, guard let, forced unwrap, nil-coalescing), функции (параметры, возвращаемые значения, inout, замыкания). Применение ООП/процедурного/мультипарадигмального подхода в Swift: классы (Class) и структуры (Struct), наследование, полиморфизм, инкапсуляция (private, fileprivate, internal, public, open), протоколы (Protocols), расширения (Extensions), вычисляемые свойства (computed properties), наблюдатели свойств (willSet, didSet). Написание кода с соблюдением функциональной парадигмы (функции высшего порядка — map, filter, reduce, compactMap, flatMap; замыкания, функциональные цепочки, неизменяемость (immutability)). Разработка консольных игр: библиотека Curses для Swift (обёртка над NCurses), управление экраном, цветом, клавиатурой, пошаговая логика, генерация карты, отрисовка, игровой цикл (на примере Rogue-like). Веб-разработка на Vapor: архитектура Vapor (Application, Routes, Controllers, Middleware), маршрутизация, обработка HTTP-запросов (Request/Response), Content API (JSON), шаблоны Leaf, асинхронность (EventLoopFuture, async/await в современных версиях). Работа с базами данных: подключение БД к Vapor (Fluent ORM), модели (Model), миграции (Migrations), отношения между моделями (parent, children, siblings), CRUD-операции. Аутентификация и авторизация в Vapor: базовые механизмы (Basic, Bearer), хэширование паролей (Bcrypt), защита маршрутов (Middleware), работа с пользователями. JWT-авторизация: структура JWT-токена, подпись (HMAC, RSA), генерация и валидация JWT в Vapor (JWTKit), настройка аутентификации через Bearer-токен, защита маршрутов, refresh-токены. Разработка мобильных приложений на iOS: архитектура (UIKit или SwiftUI), ViewController / View, сетевые запросы (URLSession, Alamofire), работа с JSON (Codable), асинхронность (GCD, async/await,	

	Combine). Создание iOS-клиента для сервера на Vapor. Добавление JWT-авторизации в iOS-приложение (хранение токена в Keychain или UserDefaults, автоматическая подстановка в заголовки запросов, обработка 401 ошибок). 12. BrickGame v3.0 на Swift Архитектура Web/Mobile приложений на Swift (бэкенд на Vapor, мобильный клиент на iOS (UIKit/SwiftUI), Web-интерфейс по выбору — Leaf или отдельный фронтенд). Обратная совместимость программных компонентов: поддержка старой версии игры и старого интерфейса при добавлении новой игры и нового интерфейса. Паттерн MVC/MVVM для игр с графическим интерфейсом (на iOS). Взаимодействие клиента и сервера (REST API, WebSocket при необходимости — Vapor WebSocket). Сохранение рекордов и прогресса (база данных на сервере через Fluent, локальное хранилище на клиенте — UserDefaults / CoreData / Keychain). 13. Интенсив AP на языке Go Основы языка Go: типы данных (int, float64, string, bool, byte, rune), переменные (var, :=), константы, операторы, ветвления (if, switch), циклы (for — единственный цикл), массивы, срезы (slices), map, строки и работа с ними (пакет strings, strconv), функции (множественное возвращение, именованные возвращаемые значения, функции как значения), работа с консолью (fmt, bufio). Применение ООП/процедурного/мультипарадигмального подхода в Go: в Go нет классических классов — вместо этого структуры (struct) и методы (func с receiver), интерфейсы (interface), композиция вместо наследования, инкапсуляция через экспортируемые (заглавная буква) и неэкспортируемые (строчная буква) идентификаторы. Написание кода с соблюдением функциональной парадигмы (функции высшего порядка, замыкания, передача функций как аргументов, работа со срезами без побочных эффектов, идиоматичный Go). Разработка консольных игр: библиотека curses для Go (например, go.etcd.io/bbolt для хранения данных или прямая работа через CGO с NCurses), управление экраном, цветом, клавиатурой, пошаговая логика, генерация карты, отрисовка, игровой цикл (на примере Rogue-like). Использование cgo для взаимодействия с C-библиотеками. Веб-разработка на Go с использованием стандартной библиотеки net/http и фреймворков (например, chi, gin, echo): маршрутизация, обработчики (handlers), middleware, работа с JSON (encoding/json), шаблоны (html/template), обработка запросов и ответов, контекст (context.Context). Работа с базами данных: подключение БД к Go-приложению (database/sql, драйверы для PostgreSQL/MySQL/SQLite), ORM-подобные библиотеки (gorm, sqlx), миграции (golang-migrate), CRUD-операции. Аутентификация и авторизация: хэширование паролей (bcrypt), управление сессиями или JWT, middleware для проверки авторизации. JWT-авторизация: структура JWT-токена, генерация и валидация JWT в Go (библиотека golang-jwt/jwt), настройка middleware для проверки Bearer-токенов, защита маршрутов, работа с refresh-токенами. Расширение возможностей веб-приложений на Go: CORS (rs/cors), обработка ошибок, Graceful shutdown, фоновые задачи (goroutines, channels), кэширование (go-cache, ristretto), метрики и логирование, тестирование (пакет testing). 14. BrickGame v3.0 на Go Архитектура Web/Mobile приложений на Go (бэкенд на net/http/gin/echo, фронтенд на HTML/CSS/JS или мобильный клиент	

	на Flutter/React Native с API на Go). Обратная совместимость программных компонентов: поддержка старой версии игры и старого интерфейса при добавлении новой игры и нового интерфейса. Паттерн MVC для игр с графическим интерфейсом (на клиенте или на сервере с Web-интерфейсом). Взаимодействие клиента и сервера (REST API, WebSocket — библиотека gorilla/websocket при необходимости). Сохранение рекордов и прогресса (база данных на сервере, локальное хранилище на клиенте: localStorage/IndexedDB для Web, SQLite для мобильного клиента).	
	Содержание практических занятий 1. Интенсив AP на языке C# Написание консольных приложений на C# с использованием основных конструкций языка. Создание иерархий классов, реализация интерфейсов, применение полиморфизма, unit-тестирование. Разработка консольной Rogue-like игры с dotnet-curses (карта, управление, базовые механики). Создание веб-приложения на ASP.NET Core MVC (страницы, формы, валидация). Подключение базы данных через Entity Framework Core, выполнение CRUD-операций. Реализация регистрации, входа, разграничения доступа по ролям. Настройка JWT-аутентификации, защита API-эндпоинтов. 2. AP2 — BrickGame v3.0 на C# Реализация игровой логики BrickGame (тетрис-лайк) на C#. Разработка пользовательского интерфейса для Web или Mobile (по выбору). Обеспечение обратной совместимости: старая игра со старым интерфейсом, новая игра с новым интерфейсом. Применение алгоритмов и структур данных для игровой логики. Сохранение рекордов и прогресса игрока. Тестирование и отладка приложения. 3. Интенсив AP на языке Java Написание консольных приложений на Java с использованием основных конструкций языка. Создание иерархий классов, реализация интерфейсов, применение функциональных возможностей (лямбды, Stream API). Разработка консольной Rogue-like игры с JCurses (карта, управление, базовые механики). Создание веб-приложения на Spring Boot (REST API, контроллеры, сервисы, репозитории). Подключение базы данных через Spring Data JPA, выполнение CRUD-операций. Реализация регистрации, входа, разграничения доступа (Spring Security). Настройка JWT-аутентификации, защита API-эндпоинтов. Создание Android-приложения как клиента для сервера (сетевые запросы, отображение данных). Добавление JWT-авторизации в Android-приложение. 4. BrickGame v3.0 на Java Реализация игровой логики BrickGame (тетрис-лайк) на Java. Разработка пользовательского интерфейса для Web или Mobile (по выбору: Spring Boot + Thymeleaf / Android-приложение / веб-фронтенд на JS + Spring API). Обеспечение обратной совместимости: старая игра со старым интерфейсом, новая игра с новым интерфейсом. Применение алгоритмов и структур данных для игровой логики. Сохранение рекордов и прогресса игрока. Тестирование и отладка приложения. 5. Интенсив AP на языке JavaScript Написание консольных скриптов на JavaScript (Node.js) с использованием основных конструкций языка. Создание иерархий классов (или объектных прототипов), применение	16

	функционального подхода (лямбды, замыкания, чистые функции). Разработка консольной Rogue-like игры с библиотекой blessed (карта, управление персонажем, базовые игровые механики) на Node.js. Создание серверного приложения на Nest.js (REST API, контроллеры, сервисы, модули). Подключение базы данных через TypeORM/Prisma, выполнение CRUD-операций. Реализация регистрации, входа, разграничения доступа (JWT + Passport). Создание клиентского приложения на Vue.js с маршрутизацией и управлением состоянием, подключение к серверному API (Axios). Добавление JWT-авторизации на клиенте (хранение токена, автоматическая подстановка в заголовки). Расширение серверного приложения: валидация, обработка ошибок, CORS, Swagger-документация. 6. BrickGame v3.0 на JavaScript Реализация игровой логики BrickGame (тетрис-лайк) на JavaScript (Node.js для бэкенда или чистый JS для клиента). Разработка пользовательского интерфейса для Web (Vue.js) или Mobile (React Native/Ionic) — по выбору. Обеспечение обратной совместимости: старая игра со старым интерфейсом, новая игра с новым интерфейсом. Применение алгоритмов и структур данных для игровой логики. Сохранение рекордов и прогресса игрока (серверная БД + клиентское хранилище). Тестирование и отладка приложения. 7. Интенсив АР на языке Kotlin Написание консольных приложений на Kotlin с использованием основных конструкций языка. Создание иерархий классов, применение функциональных возможностей (лямбды, scope-функции, работа с коллекциями). Разработка консольной Rogue-like игры с использованием NCurses через cinterop (карта, управление персонажем, базовые механики). Создание веб-приложения на Ktor (REST API, маршруты, плагины). Подключение базы данных через Exposed, выполнение CRUD-операций. Реализация регистрации, входа, разграничения доступа (JWT-плагин Ktor). Настройка JWT-аутентификации, защита маршрутов. Создание Android-приложения как клиента для сервера (Retrofit, корутины, ViewModel, отображение данных). Добавление JWT-авторизации в Android-приложение (хранение токена, перехватчики запросов). 8. BrickGame v3.0 на Kotlin Реализация игровой логики BrickGame (тетрис-лайк) на Kotlin (Ktor для бэкенда или чистый Kotlin/Android для клиента). Разработка пользовательского интерфейса для Web (Ktor + HTML/JS) или Mobile (Android) — по выбору. Обеспечение обратной совместимости: старая игра со старым интерфейсом, новая игра с новым интерфейсом. Применение алгоритмов и структур данных для игровой логики. Сохранение рекордов и прогресса игрока (серверная БД + клиентское хранилище). Тестирование и отладка приложения. 9. Интенсив АР на языке Python Написание консольных скриптов на Python с использованием основных конструкций языка. Создание иерархий классов, реализация магических методов, применение функционального подхода (map/filter/lambda/list comprehensions, декораторы). Разработка консольной Rogue-like игры с библиотекой curses (карта, управление персонажем, базовые игровые механики). Создание веб-приложения на Flask (REST API или полноценное веб-приложение с шаблонами). Подключение базы данных через SQLAlchemy, выполнение CRUD-операций. Реализация	

	регистрации, входа, разграничения доступа (Flask-Login или JWT). Настройка JWT-аутентификации, защита API-эндпоинтов. Добавление расширений по выбору: CORS, обработка ошибок, кэширование, документация API. 10. BrickGame v3.0 на Python Реализация игровой логики BrickGame (тетрис-лайк) на Python. Разработка пользовательского интерфейса для Web (Flask + Jinja2/HTML/CSS/JS) или Mobile (Kivy / Flet / Beeware) — по выбору. Обеспечение обратной совместимости: старая игра со старым интерфейсом, новая игра с новым интерфейсом. Применение алгоритмов и структур данных для игровой логики. Сохранение рекордов и прогресса игрока (серверная БД + клиентское хранилище). Тестирование и отладка приложения. 11. Интенсив AP на языке Swift Написание консольных приложений на Swift с использованием основных конструкций языка. Создание иерархий классов и структур, реализация протоколов, расширений, применение функционального подхода (map/filter/reduce, замыкания). Разработка консольной Rogue-like игры с использованием Curses для Swift (карта, управление персонажем, базовые механики). Создание веб-приложения на Vapor (REST API, маршруты, контроллеры, middleware). Подключение базы данных через Fluent ORM, выполнение CRUD-операций, миграции. Реализация регистрации, входа, разграничения доступа (Basic/Bearer аутентификация, Bcrypt). Настройка JWT-аутентификации, защита маршрутов. Создание iOS-приложения как клиента для сервера (URLSession, Codable, async/await, отображение данных). Добавление JWT-авторизации в iOS-приложение (хранение токена в Keychain, автоматическая подстановка в заголовки). 12. BrickGame v3.0 на Swift Реализация игровой логики BrickGame (тетрис-лайк) на Swift (Vapor для бэкенда или чистый Swift/iOS для клиента). Разработка пользовательского интерфейса для Web (Vapor + Leaf) или Mobile (iOS на UIKit/SwiftUI) — по выбору. Обеспечение обратной совместимости: старая игра со старым интерфейсом, новая игра с новым интерфейсом. Применение алгоритмов и структур данных для игровой логики. Сохранение рекордов и прогресса игрока (серверная БД + клиентское хранилище). Тестирование и отладка приложения. 13. Интенсив AP на языке Go Написание консольных программ на Go с использованием основных конструкций языка. Создание структур с методами, реализация интерфейсов, применение функционального подхода (функции как значения, замыкания). Разработка консольной Rogue-like игры с использованием Curses через CGO (карта, управление персонажем, базовые механики). Создание веб-приложения на Go (net/http + chi/gin) (REST API, маршруты, обработчики, middleware). Подключение базы данных через database/sql или gorm, выполнение CRUD-операций, миграции. Реализация регистрации, входа, разграничения доступа (JWT + middleware). Настройка JWT-аутентификации, защита API-эндпоинтов. Добавление расширений по выбору: CORS, Graceful shutdown, фоновые задачи, логирование, кэширование. 14. BrickGame v3.0 на Go Реализация игровой логики BrickGame (тетрис-лайк) на Go. Разработка пользовательского интерфейса для Web (Go templates + HTML/CSS/JS) или Mobile (создание отдельного мобильного	

	клиента, взаимодействующего с Go-сервером по API) — по выбору. Обеспечение обратной совместимости: старая игра со старым интерфейсом, новая игра с новым интерфейсом. Применение алгоритмов и структур данных для игровой логики. Сохранение рекордов и прогресса игрока (серверная БД + клиентское хранилище). Тестирование и отладка приложения.	
	СРС Синтаксис и основы языка — переменные, типы данных, операторы, ветвления, циклы, массивы/списки/срезы, строки, функции, работа с консолью. Объектно-ориентированное программирование — классы/структуры, наследование/композиция, полиморфизм, инкапсуляция, интерфейсы/протоколы, магические/специальные методы. Функциональное программирование — функции высшего порядка, лямбды/замыкания, map/filter/reduce, иммутабельность, работа с коллекциями. Консольная разработка игр (curses/NCurses/blessed/JCurses) — инициализация экрана, управление вводом, отрисовка, игровой цикл, генерация карты, пошаговая логика (Rogue-like). Веб-разработка (бэкенд) — маршрутизация, контроллеры/обработчики, middleware, работа с HTTP-запросами и ответами, JSON, шаблоны. Фреймворки / библиотеки для веба — ASP.NET (C#), Spring (Java), Nest.js (JS), Ktor (Kotlin), Flask (Python), Vapor (Swift), net/http/gin/chi (Go). Базы данных и ORM/ODM — подключение БД, модели, миграции, CRUD-операции, связи между таблицами (Entity Framework, Spring Data JPA, TypeORM/Prisma, Exposed, SQLAlchemy, Fluent, Gorm/database/sql). Аутентификация и авторизация — хэширование паролей, сессии/cookie, роли и права доступа, middleware для защиты маршрутов. JWT-авторизация — структура токена, генерация и валидация, Bearer-токены, защита API, refresh-токены. Мобильная разработка (Android / iOS) — архитектура (Activity/Fragment/ViewModel или UIKit/SwiftUI), сетевые запросы (Retrofit/URLSession), работа с JSON, асинхронность (корутины/GCD/Combine), хранение токенов (SharedPreferences/Keychain). Клиент-серверное взаимодействие — создание клиента (Android/iOS/Vue.js), подключение к серверному API, авторизация через JWT, обработка ошибок, обновление токенов. Архитектура приложений и обратная совместимость — паттерны (MVC/MVVM), поддержка старой и новой версий игры и интерфейса, сохранение данных (БД + локальное хранилище), тестирование.	12
Промежуточная аттестация	*Экзамен*	2
Модуль 5. Инфраструктурные решения и управление данными		32
Тема 5.1 *Системное администрирование и введение в DevOps*	*Содержание теоретических занятий* 1. Linux Выбор дистрибутива (Ubuntu/Debian/CentOS/RHEL). Установка Linux на виртуальную машину (VirtualBox/VMware/QEMU) Базовые команды Linux: навигация по файловой системе (cd, ls, pwd), работа с файлами и каталогами (cp, mv, rm, mkdir, touch), просмотр файлов (cat, less, head, tail), права доступа (chmod, chown), процессы (ps, top, kill), управление пакетами (apt, yum, dnf). Обновление системы и пакетов. Настройка пользователей и групп (useradd, usermod, groupadd, sudo). Настройка SSH (sshd, ключи доступа). Планировщик задач (cron, systemd timers). 2. Linux Network Сетевые интерфейсы (eth0, ens33, lo). Настройка IP-адресации	5

	(статический IP, DHCP). Конфигурационные файлы сети (/etc/network/interfaces, Netplan). Маршрутизация (route, ip route). DNS-настройки (/etc/resolv.conf, systemd-resolved). Сетевые утилиты (ping, traceroute, netstat, ss, tcpdump, nmap). Брандмауэр (iptables, ufw, nftables). Настройка сетевых мостов и NAT в виртуальных машинах. Взаимодействие между несколькими виртуальными машинами. 3. SimpleDocker Основы Docker: образы (images), контейнеры (containers), Dockerfile, Docker Hub. Docker Compose для многоконтейнерных приложений. CI/CD: концепция непрерывной интеграции и непрерывной доставки. Этапы CI/CD: сборка → тестирование → упаковка → развёртывание. Пример проекта (например, SimpleBashUtils). Автоматизация сборки через Makefile и скрипты. 4. CICD Интеграция всех предыдущих тем в единый CI/CD-пайплайн. Настройка CI/CD-сервера (GitLab CI, GitHub Actions, Jenkins). Пайплайн: линтинг → сборка → unit-тесты → интеграционные тесты → сборка Docker-образа → деплой. Инфраструктура как код (IaC). Оркестрация контейнеров (введение в Kubernetes / Docker Swarm).	
	Содержание практических занятий 1. Linux Установка Linux на виртуальную машину. Настройка сети (статический IP, DNS). Создание и настройка пользователей, выдача прав sudo. Настройка SSH-доступа по ключу. Настройка автоматических задач через cron. Обновление системы и установка необходимых пакетов. 2. Linux Network Настройка статического IP на виртуальной машине. Настройка маршрутизации между двумя виртуальными машинами. Настройка DNS-резолвера. Настройка брандмауэра (ufw/iptables) для ограничения доступа. Использование tcpdump для анализа трафика. Проверка сетевой связности между ВМ. 3. SimpleDocker Написание Dockerfile для проекта на C. Сборка Docker-образа. Запуск контейнера и проверка работоспособности. Настройка автоматической сборки и тестирования в контейнере. Упаковка образа и загрузка в реестр (Docker Hub). Развёртывание контейнера на тестовом окружении. 4. CICD Настройка CI/CD на выбранной платформе (GitLab CI / GitHub Actions). Написание конфигурации пайплайна (YAML). Интеграция линтера, сборки, тестов, Docker-сборки в один пайплайн. Автоматический деплой контейнера на сервер после успешных тестов. Настройка триггеров (push, merge request, по расписанию).	7
	СРС Установка Linux на виртуальную машину, базовая настройка (пользователи, SSH, права, пакеты, обновления). Настройка сети в Linux (статический IP, маршрутизация, DNS, брандмауэр, сетевые утилиты). Основы Bash: написание скриптов (переменные, циклы, условия, функции, обработка ошибок). Основы Docker: Dockerfile, сборка образов, запуск контейнеров, Docker Compose, Docker Hub. Разработка CI/CD пайплайна (сборка → тесты → Docker → деплой) на GitHub Actions или GitLab CI. Интеграция проекта (например, SimpleBashUtils) в CI/CD: линтинг, тесты, упаковка в Docker. Основы мониторинга: системные утилиты (htop, iotop, netstat, df),	5

	знакомство с Prometheus + Grafana. Комплексная практика: объединение Linux, Bash, Docker, CI/CD и мониторинга в экзаменационных заданиях.	
Тема 5.2 *Базы данных (SQL)*	*Содержание теоретических занятий* 1. SQL Интенсив Реляционная модель данных: таблицы, строки, столбцы, первичные ключи (PRIMARY KEY), внешние ключи (FOREIGN KEY), связи между таблицами (один-ко-многим, многие-ко-многим, один-к-одному), нормализация. Создание и удаление таблиц: CREATE TABLE (типы данных: INT, VARCHAR, DATE, DECIMAL, BOOLEAN и др., ограничения: NOT NULL, UNIQUE, DEFAULT, CHECK), DROP TABLE, ALTER TABLE (добавление/удаление/изменение столбцов). Вставка данных: INSERT INTO ... VALUES, INSERT INTO ... SELECT, вставка нескольких строк за раз. Выборка данных: SELECT (выбор всех или конкретных столбцов), SELECT DISTINCT (уникальные значения), SELECT TOP / LIMIT (ограничение количества строк). Фильтрация данных: WHERE (операторы: =, <>, >, <, >=, <=, BETWEEN, LIKE, IN, IS NULL, AND, OR, NOT). Упорядочивание данных: ORDER BY (ASC, DESC), сортировка по нескольким столбцам. Группировка данных: GROUP BY, агрегационные функции (MIN, MAX, AVG, SUM, COUNT), HAVING (фильтрация после группировки), COUNT(DISTINCT ...). Объединение данных: JOIN (INNER JOIN), LEFT JOIN, RIGHT JOIN, CROSS JOIN, объединение по нескольким условиям, самообъединение (self join). Математические и строковые функции: ROUND, CEIL, FLOOR, ABS, POWER, LENGTH, UPPER, LOWER, SUBSTRING, CONCAT, COALESCE, CAST. Обновление данных: UPDATE ... SET ... WHERE (с условием и без), обновление нескольких столбцов. Удаление данных: DELETE FROM ... WHERE, TRUNCATE TABLE (быстрое удаление всех строк), отличие DELETE от TRUNCATE и DROP.	4
	Содержание практических занятий 1. SQL Интенсив Создание базы данных и таблиц с ограничениями и связями (например, «Пользователи — Заказы — Товары»). Наполнение таблиц тестовыми данными через INSERT. Написание запросов на выборку с фильтрацией (WHERE), сортировкой (ORDER BY) и ограничением (LIMIT). Использование агрегатных функций и группировки (GROUP BY, HAVING). Написание запросов с различными типами JOIN (INNER, LEFT, RIGHT, CROSS) для объединения данных из нескольких таблиц. Применение математических и строковых функций в запросах. Обновление данных с помощью UPDATE (по условию, массовое обновление). Удаление данных через DELETE (по условию, удаление связанных данных с учётом внешних ключей). Написание сложных запросов, комбинирующих фильтрацию, группировку, JOIN и функции. Тестирование запросов на корректность и производительность.	5
	СРС Изучение реляционной модели данных: первичные и внешние ключи, нормальные формы (1НФ, 2НФ, 3НФ), типы связей между таблицами. Освоение синтаксиса создания и модификации таблиц (CREATE TABLE, ALTER TABLE, DROP TABLE) с ограничениями (NOT NULL, UNIQUE, CHECK, DEFAULT). Практика написания запросов на выборку: SELECT, WHERE, ORDER BY, LIMIT, DISTINCT, агрегатные функции, GROUP BY, HAVING. Практика объединения таблиц: INNER JOIN, LEFT JOIN,	4

	RIGHT JOIN, CROSS JOIN, самообъединение, объединение по нескольким условиям. Освоение встроенных функций SQL: математические (ROUND, CEIL, ABS), строковые (CONCAT, SUBSTRING, LENGTH), работа с датами, COALESCE, CAST. Практика модификации данных: INSERT (одиночная и массовая вставка), UPDATE (обновление с условием и без), DELETE и TRUNCATE, каскадные операции.	
Промежуточная аттестация	*Экзамен*	2
Модуль 6. Старт бизнеса		56
Тема 6.1 *Что такое Lean startup*	*Содержание теоретических занятий* 1. Что такое Lean startup Происхождение Lean Startup (Эрик Рис, проблемы традиционного подхода к запуску продуктов). Ключевые принципы: Build → Measure → Learn (цикл обратной связи). MVP (Minimum Viable Product) — минимально жизнеспособный продукт: что это, зачем нужен, примеры. Pivot (разворот) и Persevere (продолжение): когда менять стратегию, а когда оставаться на курсе. Инновационная бухгалтерия: как измерять прогресс в условиях неопределённости. Отличие Lean Startup от традиционного бизнес-планирования	4
	Содержание практических занятий 1. Что такое Lean startup Анализ кейсов успешных и провальных стартапов через призму Lean Startup. Определение MVP для выбранной идеи (минимальный набор функций для проверки гипотезы). Построение цикла Build → Measure → Learn для собственного проекта. Разработка гипотез о ценности и росте продукта.	5
	СРС Прочитать книгу Эрика Риса «Lean Startup» (или краткое содержание ключевых глав). Разобрать 3 примера стартапов, которые использовали или должны были использовать Lean Startup. Сформулировать MVP для своей идеи (или выбранного кейса) и обосновать выбор функций. Подготовить схему цикла Build → Measure → Learn для конкретного продукта	4
Тема 6.2 *Как быстро тестировать гипотезы с помощью Custdev*	*Содержание теоретических занятий* 1. Как быстро тестировать гипотезы с помощью Custdev Custdev (Customer Development) — развитие через клиентов: концепция Стива Бланка. Отличие Custdev от традиционных маркетинговых исследований. Типы гипотез: гипотезы о проблеме, гипотезы о решении, гипотезы о каналах привлечения, ценовые гипотезы. Скрипты интервью: как задавать вопросы без искажений (не наводящие, не продающие, не «нравится ли вам»). Поиск респондентов: где и как находить целевую аудиторию для интервью. Обработка результатов: качественные и количественные методы, выявление паттернов. Ошибки при Custdev: интервью с друзьями, вопросы о будущем, игнорирование «молчащих» респондентов.	4
	Содержание практических занятий 1. Как быстро тестировать гипотезы с помощью Custdev Формулировка гипотез для собственного проекта (3–5 гипотез). Составление скрипта интервью (10–15 вопросов). Проведение минимум 5 реальных интервью с потенциальными клиентами. Анализ полученных данных: подтверждение или опровержение гипотез.	5
	СРС Изучить книгу Стива Бланка «The Four Steps to the Epiphany» (Customer Development). Подготовить шаблон скрипта интервью	5

	для своей целевой аудитории. Найти 10 потенциальных респондентов (в соцсетях, профильных сообществах, через личные связи). Провести интервью, записать ответы и сделать выводы по каждой гипотезе.	
Тема 6.3 *Как оценивать рынок и избежать типичных ошибок при оценке*	*Содержание теоретических занятий* 1. Как оценивать рынок и избежать типичных ошибок при оценке TAM, SAM, SOM: что это и как считать (Total Addressable Market, Serviceable Available Market, Serviceable Obtainable Market). Методы оценки рынка: top-down (сверху вниз), bottom-up (снизу вверх), сравнительный метод. Источники данных для оценки рынка (открытая статистика, отчёты, опросы, экспертные оценки). Типичные ошибки: переоценка рынка, игнорирование конкурентов, путаница TAM и SOM, экстраполяция без учёта ограничений. Анализ конкурентов: прямые, косвенные, потенциальные (карта позиционирования). Тренды и макроэкономические факторы, влияющие на рынок	3
	Содержание практических занятий 1. Как оценивать рынок и избежать типичных ошибок при оценке Расчёт TAM, SAM, SOM для выбранной ниши. Поиск и анализ 3–5 отчётов по рынку (например, от Gartner, Statista, государственная статистика). Составление карты конкурентов (ось X — цена, ось Y — качество/функциональность). Выявление основных рисков и возможностей на рынке.	4
	СРС Рассчитать TAM, SAM, SOM для своей идеи (с обоснованием цифр и источников). Провести анализ 3 прямых конкурентов (продукт, цена, аудитория, каналы привлечения). Определить 3 типичные ошибки оценки рынка на примере реальных провалившихся стартапов. Подготовить краткий аналитический отчёт по рынку (ёмкость, динамика, тренды, барьеры входа)	3
Тема 6.4 *Как масштабировать бизнес за пределы региона*	*Содержание теоретических занятий* 1. Как масштабировать бизнес за пределы региона Модели масштабирования: органический рост, экспансия в новые регионы, франчайзинг, партнёрства, лицензирование. Оценка готовности к масштабированию: юнит-экономика, операционные процессы, команда, продукт. Адаптация продукта/услуги под новый рынок: культурные, языковые, юридические, логистические особенности. Выбор первого региона для экспансии: критерии (размер рынка, близость, конкуренция, регуляторика). Юридические аспекты: регистрация компании, налоги, лицензии, защита интеллектуальной собственности. Управление удалённой командой и построение филиальной сети. Риски масштабирования: потеря контроля, размывание культуры, кассовые разрывы, валютные риски.	3
	Содержание практических занятий 1. Как масштабировать бизнес за пределы региона Выбор региона для гипотетической экспансии (обоснование критериями). Разработка плана адаптации продукта под новый рынок (изменения в продукте, маркетинге, ценообразовании). Оценка юнит-экономики до и после масштабирования. Составление roadmap масштабирования (по шагам, с сроками и ресурсами).	3
	СРС Изучить кейс масштабирования известной компании (например, Яндекс, Ozon, Dodo Pizza, Lamoda) за пределы России/региона. Составить список из 5 критериев выбора региона для своей идеи. Рассчитать бюджет на первый этап масштабирования (регистрация,	3

	адаптация, команда, маркетинг). Подготовить план по найму и управлению удалённой командой в новом регионе.	
Тема 6.5 *Как начать бизнес с франшизой*	*Содержание теоретических занятий* 1. Как начать бизнес с франшизой Франшиза: что это такое (договор коммерческой концессии), стороны (франчайзи и франчайзор). Виды франшиз: товарные, производственные, сервисные, бизнес-формат (самый полный). Структура франшизы: паушальный взнос, роялти, маркетинговые отчисления, инвестиции. Что входит во франшизу: бренд, технологии, обучение, поддержка, стандарты, ПО, поставщики. Плюсы франшизы: готовая бизнес-модель, снижение рисков, поддержка, узнаваемость. Минусы франшизы: платежи, ограничения, зависимость от франчайзора, риск репутации. Как выбрать франшизу: проверка франчайзора (открытость, раскрытие финансов, судебные споры, отзывы франчайзи). Юридическая проверка: договор франшизы, регистрация, права и обязанности.	2
	Содержание практических занятий 1. Как начать бизнес с франшизой Анализ 3 существующих франшиз (паушальный взнос, роялти, инвестиции, условия). Расчёт окупаемости франшизы (стартовые инвестиции, ежемесячные платежи, прогноз выручки). Составление вопросов к франчайзору (чек-лист из 20–30 вопросов). Сравнение «купить франшизу» vs «открыть свой бизнес с нуля» для конкретной ниши.	3
	СРС Найти 3 франшизы в интересующей нише и составить сравнительную таблицу (взнос, роялти, инвестиции, обучение, поддержка). Рассчитать примерный срок окупаемости и прибыль по одной из них. Составить список из 10 вопросов для проверки надёжности франчайзора. Прочитать договор франшизы (шаблон или реальный) и выделить ключевые риски и ограничения	3
Промежуточная аттестация	Экзамен	2
Модуль 7. Старт бизнеса		44
Тема 7.1 *Продуктовый подход и работа с гипотезам*	*Содержание теоретических занятий* 1. Продуктовый подход и работа с гипотезам Продуктовый подход vs проектный подход: в чём разница. Продукт как инструмент решения проблемы клиента. Гипотеза: что это, структура (если мы сделаем X, то получим Y, потому что Z). Виды гипотез: ценностные, ростовые, канальные, ценовые. Приоритизация гипотез (ICE, RICE, PIE — Impact, Confidence, Ease). Эксперименты как способ проверки гипотез: A/B-тесты, фейковые двери, предзаказы, прототипы. Метрики для проверки гипотез: качественные и количественные.	3
	Содержание практических занятий 1. Продуктовый подход и работа с гипотезам Формулировка 5–7 гипотез для собственного продукта по структуре «если — то — потому что». Приоритизация гипотез методом ICE или RICE. Выбор типа эксперимента для проверки топ-3 гипотез. Разработка протокола эксперимента (шаги, сроки, критерии успеха).	4
	СРС Изучить методологии приоритизации гипотез (ICE, RICE, PIE). Сформулировать 10 гипотез для выбранного продукта. Провести приоритизацию и выбрать 3 гипотезы для ближайшего эксперимента. Разработать дизайн эксперимента для одной гипотезы (включая метрики успеха)	3

Тема 7.2 *Cusdev и концепция Jobs to be Done*	*Содержание теоретических занятий* 1. Cusdev и концепция Jobs to be Done Custdev (Customer Development) как процесс непрерывного взаимодействия с клиентами. Этапы Custdev: поиск проблемы, проверка решения, проверка бизнес-модели, масштабирование. Jobs to be Done (JTBD) — концепция «работ» клиента: что клиент пытается «сделать» (прогресс в определённом контексте). Функциональные, эмоциональные и социальные работы. JTBD-фреймворк: ситуация, мотивация, ожидаемый результат, альтернативы. Отличие JTBD от персон (personas). Проведение JTBD-интервью: как выявлять не потребности, а «работы».	3
	Содержание практических занятий 1. Cusdev и концепция Jobs to be Done Подготовка скрипта JTBD-интервью (фокус на контекст, триггеры, боли, альтернативы). Проведение минимум 3 интервью с фокусом на выявление «работ» клиента. Построение JTBD-карты для своей целевой аудитории. Формулировка ценностного предложения на основе выявленных работ. Custdev и JTBD: в чём разница и когда что применять	4
	СРС Изучить книгу Клейтона Кристенсена «Jobs to be Done» (ключевые главы). Провести 5 интервью по JTBD-методологии с записью и анализом. Составить JTBD-профиль для сегмента пользователей (3–5 ключевых работ). Сравнить подходы	3
Тема 7.3 *Анализ рынка*	*Содержание теоретических занятий* 1. Анализ рынка Структура анализа рынка: макро- (PESTEL), мезо- (отрасль), микро- (конкуренты, клиенты). PESTEL-анализ: политические, экономические, социальные, технологические, экологические, юридические факторы. Анализ отрасли: модель 5 сил Портера (угроза новых игроков, угроза заменителей, рыночная власть поставщиков, власть покупателей, интенсивность конкуренции). Сегментация рынка: критерии (географические, демографические, поведенческие, психографические). Оценка ёмкости и динамики рынка (TAM, SAM, SOM — повторение с углублением). Анализ конкурентов: матрица конкурентных преимуществ, бенчмаркинг, Mystery Shopping.	3
	Содержание практических занятий 1. Анализ рынка Проведение PESTEL-анализа для своей ниши (по 2–3 фактора на каждую категорию). Анализ отрасли через модель 5 сил Портера. Сегментация целевой аудитории (3–5 сегментов с описанием). Построение карты конкурентов (2 оси, например, цена и качество).	3
	СРС Провести PESTEL-анализ для своей идеи (с источниками данных). Нарисовать карту конкурентов (10–15 игроков) и выделить свою позицию. Оценить TAM, SAM, SOM с обоснованием (повторение с детализацией). Составить профиль 3 ключевых сегментов аудитории (потребности, поведение, каналы)	2
Тема 7.4 *Продажи через ценность*	*Содержание теоретических занятий* 1. Продажи через ценность Ценность vs цена: что такое ценность с точки зрения клиента (выгода - затраты). Value Proposition Canvas (холст ценностного предложения): профиль клиента (боли, выгоды, работы) и карта ценности (продукт, облегчение болей, создание выгод). Формулировка ценностного предложения: шаблоны (например, «Мы помогаем [клиент] [решить проблему] с помощью [уникальное	3

	решение], в отличие от [альтернатива]»). Продажи через ценность (Value Selling): как продавать, а не просто информировать. Техника SPIN-продаж: Situation (ситуация), Problem (проблема), Implication (последствие), Need-payoff (ценность решения). Возражения и работа с ними: как переводить разговор из цены в ценность. Презентация ценности: сторителлинг, кейсы, демонстрация ROI.	
	Содержание практических занятий 1. Продажи через ценность Заполнение Value Proposition Canvas для своего продукта. Формулировка ценностного предложения в 1 предложение (шаблон). Разработка скрипта SPIN-вопросов для своей целевой аудитории. Подготовка 3-минутной презентации ценности продукта (питч)	3
	СРС Изучить книгу «SPIN-продажи» (Ник Рэкхем) или краткое содержание. Заполнить Value Proposition Canvas для своего продукта (в деталях). Написать 3 варианта ценностного предложения и выбрать лучший (тест на коллегах). Подготовить ответы на 5 типичных возражений клиентов (например, «дорого», «уже есть поставщик»).	2
Тема 7.5 *Финансовые модели и юнит-экономика*	*Содержание теоретических занятий* 1. Финансовые модели и юнит-экономика Финансовая модель: что это, зачем нужна (прогнозирование, оценка инвестиций, управление). Структура финансовой модели: доходы, расходы, прибыль, денежный поток, инвестиции. Юнит-экономика (Unit Economics): экономика одной единицы продукта/клиента. Ключевые метрики юнит-экономики: CAC (Customer Acquisition Cost), LTV (Lifetime Value), ARPU (Average Revenue Per User), Gross Margin, Contribution Margin. Формулы: CAC = затраты на маркетинг и продажи / количество новых клиентов; LTV = ARPU × средний срок жизни клиента × маржинальность. Хорошая юнит-экономика: LTV > CAC (обычно LTV / CAC > 3). Построение P&L (отчёт о прибылях и убытках) для стартапа. Основные показатели: выручка, себестоимость, операционные расходы (R&D, маркетинг, G&A), EBITDA, чистая прибыль. Cash Runway (длина взлётной полосы): на сколько месяцев хватит денег. Break-even point (точка безубыточности).	2
	Содержание практических занятий 1. Финансовые модели и юнит-экономика Расчёт CAC и LTV для своего продукта (или реального бизнеса). Оценка соотношения LTV / CAC (хорошо ли оно). Построение упрощённой финансовой модели на 12 месяцев (доходы, расходы, прибыль, денежный поток). Расчёт точки безубыточности и Cash Runway.	2
	СРС Изучить метрики юнит-экономики и формулы расчёта. Рассчитать CAC, LTV, ARPU для своего продукта (с обоснованием допущений). Построить финансовую модель в Excel/Google Sheets на 12–24 месяца (3 сценария: пессимистичный, реалистичный, оптимистичный). Рассчитать, сколько нужно инвестиций для выхода на точку безубыточности.	2
Промежуточная аттестация	*Экзамен*	2
Модуль 8. Продуктовый подход в бизнесе		44
Тема 8.1 *Личное видение. Устойчивая мотивация*	*Содержание теоретических занятий* 1. Личное видение. Устойчивая мотивация основателя Личное видение: что это, зачем нужно основателю. Отличие	2

основателя	видения от миссии и целей (Vision vs Mission vs Goals). Компоненты видения: что мы создаём, для кого, как меняем мир, какими будем через 5–10 лет. Устойчивая мотивация: внутренняя vs внешняя мотивация. Источники мотивации основателя: смысл, автономия, мастерство, признание, влияние. Выгорание: признаки, причины, профилактика. Ритуалы и привычки для поддержания энергии и фокуса. Примеры: как известные основатели формулировали своё видение (Стив Джобс, Илон Маск, основатели локальных компаний).	
	Содержание практических занятий 2. Личное видение. Устойчивая мотивация основателя Формулировка личного видения (1–2 абзаца) на 5–10 лет. Определение своей миссии как основателя. Выявление своих ключевых источников мотивации. Разработка ритуалов для поддержания устойчивой мотивации.	2
	СРС Написать личное видение (текст) и пересмотреть его через неделю. Изучить биографии 2–3 основателей и выделить их источники мотивации. Составить список действий, которые восстанавливают энергию (на каждый день). Пройти тест на выявление склонности к выгоранию и составить план профилактики.	2
Тема 8.2 *Технологическое предпринимательство. Основные понятия*	*Содержание теоретических занятий* 1. Технологическое предпринимательство. Основные понятия Технологическое предпринимательство: определение, отличие от традиционного. Технологический стартап: высокий риск, высокий рост, масштабируемость, цифровые продукты. Инновация: что это, типы инноваций (продуктовые, процессные, маркетинговые, организационные). Технологическая платформа: что это, примеры (iOS, Android, AWS, Telegram Bot API). Deep tech vs цифровые продукты: отличия. Жизненный цикл технологического стартапа (pre-seed → seed → early → growth → exit). Роли в технологическом стартапе: основатель, CTO, CPO, тимлид, разработчики. Интеллектуальная собственность: патенты, ноу-хау, коммерческая тайна.	2
	Содержание практических занятий 1. Технологическое предпринимательство. Основные понятия Анализ своего проекта: является ли он технологическим стартапом (по критериям). Определение типа инновации в своём продукте. Идентификация технологической платформы (своей или партнёрской). Составление карты стейкхолдеров своего технологического проекта.	2
	СРС Изучить отличие технологического стартапа от малого бизнеса (таблица сравнения). Найти 5 примеров технологических стартапов и определить их тип инновации. Составить список технологических трендов, которые могут повлиять на свою нишу. Разобраться с основами защиты интеллектуальной собственности в своей области.	2
Тема 8.3 *Стратегия подрыва отраслей с помощью технологий*	*Содержание теоретических занятий* 1. Стратегия подрыва отраслей с помощью технологий Концепция подрывных инноваций (Disruptive Innovation) Клейтона Кристенсена. Поддерживающие vs подрывные инновации (Sustaining vs Disruptive). Механизм подрыва: новый игрок заходит с низкого сегмента или нового рынка, лидеры игнорируют → со временем улучшается и вытесняет лидеров. Примеры: Netflix vs Blockbuster, цифровые камеры vs плёнка, такси vs Uber. Почему лидеры рынка проигрывают: инерция, фокус на лучших клиентах,	2

	страх каннибализации. Технологии как драйвер подрыва: AI, блокчейн, IoT, биотехнологии. Условия для подрыва: быстрое улучшение технологии, асимметричная мотивация, низкая маржинальность для лидера.	
	Содержание практических занятий 1. Стратегия подрыва отраслей с помощью технологий Анализ отрасли, в которой работает студент: есть ли потенциал для подрыва. Выявление «слепых зон» лидеров рынка (что они игнорируют). Разработка гипотезы подрыва: с какого сегмента можно зайти. Анализ 1–2 примеров подрыва в своей или смежной отрасли.	3
	СРС Прочитать главы Кристенсена о подрывных инновациях (краткое содержание). Проанализировать 3 примера подрывных инноваций и выделить этапы подрыва. Нарисовать график «эволюция технологии vs требования рынка» для своей ниши. Сформулировать, как можно подорвать лидера в своей отрасли (гипотеза).	2
Тема 8.4 *Подрывные инновации или как стартапу победить лидера рынка*	*Содержание теоретических занятий* 1. Подрывные инновации или как стартапу победить лидера рынка Стратегии подрыва: low-end disruption (дешёвый сегмент) и new-market disruption (новый рынок). Этапы подрыва: вход → скрытый рост → переключение → вытеснение. Барьеры для лидера: бизнес-модель, каналы продаж, культура, KPI. Что делает стартап: меньшая функциональность, но дешевле/удобнее/проще. Точки входа: недопотреблённые клиенты, переобслуженные клиенты, новые сценарии. Асимметричная война: игра по правилам, которые невыгодны лидеру. Примеры побед: Uber vs такси, Airbnb vs отели, Zoom vs Cisco WebEx. Риски для стартапа: лидер копирует, регуляторика, недостаточное финансирование.	2
	Содержание практических занятий 1. *Подрывные инновации или как стартапу победить лидера рынка* Выбор лидера в своей отрасли и анализ его слабых мест. Определение стратегии подрыва (low-end или new-market) для своего стартапа. Разработка MVP для проверки подрывной гипотезы. Построение карты «что лидер даёт лишнее, что не даёт нужного».	3
	СРС Составить профиль лидера рынка (доля, аудитория, бизнес-модель, слабые места). Определить 3 сегмента, которые лидер игнорирует или переобслуживает. Разработать стратегию подрыва для своего стартапа (1–2 страницы). Изучить кейс успешного подрыва из своей отрасли (или максимально близкой).	2
Тема 8.5 *Стратегия голубого океана*	*Содержание теоретических занятий* 1. Стратегия голубого океана Концепция «Голубой океан» (Blue Ocean Strategy): Ким и Моборн. Красный океан: жёсткая конкуренция, ценовые войны, ограниченный рынок. Голубой океан: новая ниша, отсутствие конкуренции, создание нового спроса. Инструменты: стратегическая канва (Strategic Canvas), четыре действия (убрать, уменьшить, повысить, создать), матрица «снижение-создание». Кривая ценности (Value Curve): сравнение с конкурентами. Пути создания голубого океана: посмотреть на альтернативные отрасли, стратегические группы, цепочку покупателей, дополнительные услуги, функционально-эмоциональный профиль, тренды. Примеры: Cirque du Soleil (без животных и звёзд), Nintendo Wii (не за графику), Yellow Tail (простое вино).	2

	Содержание практических занятий 1. Стратегия голубого океана Построение стратегической канвы для своей отрасли. Заполнение матрицы «убрать — уменьшить — повысить — создать». Определение пути создания голубого океана для своего продукта. Формулировка новой кривой ценности (чем отличаемся от конкурентов).	2
	СРС Изучить книгу «Стратегия голубого океана» (ключевые главы). Построить стратегическую канву для своей отрасли (минимум 5–7 факторов). Заполнить матрицу «убрать, уменьшить, повысить, создать» (4–6 пунктов в каждой). Сформулировать стратегию голубого океана для своего стартапа (1 абзац).	2
Тема 8.6 *Факторы успеха стартапа.* *Команда основателей*	*Содержание теоретических занятий* 1. Факторы успеха стартапа. Команда основателей Ключевые факторы успеха стартапов по исследованиям (CB Insights, Startup Genome). Топ-5 причин провала: нет рыночной потребности, проблемы с командой, конкуренты, цена/издержки, плохой продукт. Роль команды основателей: статистика (команды из 2–3 основателей успешнее одиночек). Комплементарность навыков: бизнес + технологии + продукт (или «hustler, hacker, hipster»). Распределение ролей и ответственности, урегулирование конфликтов. Equity (доли): как делить, вестинг (vesting), cliff, акционерное соглашение. Найм первых сотрудников: культура, ценности, проверка на соответствие. Корпоративная культура в стартапе: ценности, нормы, традиции.	2
	Содержание практических занятий 1. Факторы успеха стартапа. Команда основателей Оценка своей команды (если есть) или идеального профиля основателей для своей идеи. Распределение ролей и расчёт долей (proposed equity split). Написание акционерного соглашения (основные пункты). Разработка профиля первых 3 сотрудников (кого нанимать в первую очередь).	2
	СРС Изучить исследование CB Insights о причинах провала стартапов. Составить профиль идеальной команды из 3 основателей для своей идеи. Рассчитать вестинг (4 года + cliff 1 год) для своей команды. Написать список культурных ценностей своего стартапа (5–7 пунктов).	1
Тема 8.7 *Клиент и его потребности.* *Востребованный продукт*	*Содержание теоретических занятий* 1. Клиент и его потребности. Востребованный продукт. Product-Market Fit (PMF) — соответствие продукта рынку (Marc Andreesen). Признаки PMF: высокий retention, органический рост, сарафанное радио, клиенты расстраиваются, когда продукт недоступен. Как измерить PMF: опрос (насколько расстроитесь, если продукт исчезнет) — 40% порог. Путь к PMF: итерации, поиск каналов, сегментирование. Методы выявления потребностей: Custdev, наблюдение, анализ данных. Kano-модель: базовые потребности (must-be), линейные (performance), восторгающие (delighters). Jobs to be Done (повторение и углубление): что клиент нанимает продукт сделать. Customer Journey Map (CJM): карта пути клиента от осознания проблемы до лояльности. Боли и точки трения клиента на каждом этапе.	2
	Содержание практических занятий 1. Клиент и его потребности. Востребованный продукт Проведение опроса на PMF (вопрос «как сильно расстроитесь, если продукт исчезнет?»). Построение Customer Journey Map для своего	2

	продукта (5–7 этапов). Выявление базовых, линейных и восторгающих потребностей (Kano). Формулировка, как достичь PMF (гипотезы и план итераций).	
	СРС Провести опрос минимум 20 потенциальных клиентов на измерение PMF. Построить Customer Journey Map для своего продукта (с болями и возможностями). Составить Kano-диаграмму для своего продукта (по 3 потребности каждого типа). Написать план из 5 итераций по достижению Product-Market Fit	1
Промежуточная аттестация	*Экзамен*	2
Модуль 9. Азбука стартапа		18
Тема 9.1 *Выбор организационно-правовой формы и расчёт налоговой нагрузки*	*Содержание теоретических занятий* 1. Выбор организационно-правовой формы и расчёт налоговой нагрузки Физ. лицо/ИП/ООО: ответственность, отчётность, ограничения, последствия отсутвия ОПФ. 1.2. Закрепление партнерства при выборе ОПФ. Обязательные взносы ИП: сколько, когда, и как ими можно воспользоваться. Налоговые режимы: УСН «доходы», УСН «доходы минус расходы», патент, АУСН (доходы, доходы - расходы), ОСНО — формулы расчёта. Когда и сколько платить: авансовые платежи, годовой налог, страховые взносы, сроки отчётности. Как перестать путать свои деньги с деньгами бизнеса	1
	Содержание практических занятий Расчёт авансового платежа: формулы для УСН 6%, УСН 15%; вычет страховых взносов Налоговые льготы для новых и (практикующих) предпринимателей	2
	СРС Если я неправильно выберу ОПФ — какие последствия? Бизнес растёт — когда и как перейти на другой налоговый режим. Если не получится начать бизнес — как закрыть бизнес без последствий	2
Тема 9.2 *Налоговая модель и управляемость бизнеса с первого дня*	*Содержание теоретических занятий* 1. Налоговая модель и управляемость бизнеса с первого дня Виды дейятельности. Как выбрать код деятельности (ОКВЭД): логика классификации, основной vs дополнительный, лицензирование. Третьи лица при регистрации ЮЛ (банки). Коммерческая осмотрительность - какие сервисы упростят жизнь: Госуслуги, ЛК налоговой, электронная подпись, полезные сервисы (Прозрачный бизнес, Рассчет патента, Для уплаты и рассчета страховых взносов - показать, как ими пользоваться). Первые шаги после регистрации: анализ рынка и выбор поставщиков, сервисы для проверки контрагентов. Типология договоров: с клиентом (оферта), с подрядчиком (ГПХ), трудовой, NDA — когда какой. Обязательные элементы в договоре: предмет, стоимость, сроки, ответственность, порядок расторжения. Типовые ошибки: нечёткие сроки, подмена трудового договора ГПХ, отсутствие порядка приёмки. Как принимать оплату по закону: касса, терминал, переводы. Пришло письмо из налоговой — как понять и как ответить. Налоговый контроль и как с ним работать: права, обязанности, алгоритм действий. Когда нужен бухгалтер - когда справитесь сами, а когда пора делегировать	1
	Работа с базой знаний в чат-боте.	2
	СРС	0
Тема 9.3 *Финансовое управление бизнесом*	*Содержание теоретических занятий* 1. Финансовое управление бизнесом Почему бизнес «не управляется». Разрыв между бухгалтерией, налогами и реальными деньгами Почему прибыль ≠ деньги.	1

	Кассовые разрывы: откуда они берутся. Ошибки учета на старте, которые ломают систему ". "Финансовая модель бизнеса. Доход, расход, прибыль, маржа. Постоянные и переменные расходы. Точка безубыточности. Как считать рентабельность. Ошибка «работаю много — зарабатываю мало».	
	Содержание практических занятий 1. Финансовое управление бизнесом Налоговая модель как инструмент управления. Когда УСН перестает быть выгодной. Признаки необходимости перехода на ОСНО. Работа с НДС: когда это плюс, а не минус. Ошибки “оптимизации”, которые приводят к доначислениям. Деньги в бизнесе: контроль и безопасность. Касса, расчётный счёт, подотчёт “Чёрная касса” — чем это заканчивается. Дробление денежных потоков. Как налоговая видит движение денег. Типовые схемы доначислений.	2
	СРС	0
Тема 9.4 Масштабирование и защита бизнеса	*Содержание теоретических занятий* 1. Масштабирование и защита бизнеса Дробление бизнеса: где грань. Что такое дробление с точки зрения ФНС. Признаки искусственного разделения. Связанные лица и аффилированность. Когда дробление допустимо. Работа с персоналом и подрядчиками. Сотрудники vs самозанятые vs ИП Переквалификация в трудовые отношения. Как налоговая доказывает фиктивность. Безопасные модели сотрудничества Ошибки договоров.	2
	Содержание практических занятий 1. Масштабирование и защита бизнеса Вывод денег и личная стратегия собственника. Дивиденды, зарплата, займы. Как собственник “забирает деньги”. Ошибки, которые приводят к доначислениям. Баланс между бизнесом и личными финансами.	2
	СРС Налоговые проверки и защита. Как выбирают на проверку. Камеральные vs выездные. Требования ФНС: как правильно отвечать. Ошибки коммуникации с налоговой. Подготовка к проверке"	1
Промежуточная аттестация	*Экзамен*	2
Итоговая аттестация	*Экзамен*	4
Итого		**248**

Промежуточная аттестация - экзамен

Итоговая аттестация - экзамен

2.3. Оценочные материалы

Модуль1. Основы IT-инфраструктуры

Задание 1. Основы Linux и работа с файловой системой

В домашней директории создайте структуру каталогов и выполните операции с файлами согласно списку. Результат оформите в виде отчёта с командами и скриншотами.

Список действий:

Создать структуру: ~/exam/projects/, ~/exam/backups/, ~/exam/temp/

В projects создать файлы: readme.txt, notes.txt, todo.txt

Скопировать todo.txt в backups

Переместить notes.txt в temp

Установить на readme.txt права 640

Найти все .txt файлы в домашней директории

Эталон выполнения

Отчёт со всеми командами (например, mkdir, touch, cp, mv, chmod, find)

Скриншот вывода ls -la ~/exam

Скриншот вывода find ~ -name "*.txt" 2>/dev/null

Критерии оценки

Что проверяется	Балл
Все каталоги созданы	1
Все файлы созданы	1
Копирование и перемещение выполнены верно	1
Права 640 установлены	1
Поиск файлов выполнен	1
Итого	5

Задание 2. Bash-скрипт для резервного копирования

Напишите Bash-скрипт backup.sh, который принимает два аргумента: исходную директорию и директорию для резервного копирования.

Требования к скрипту:

Проверить, существует ли исходная директория. Если нет — вывести ошибку.

Создать директорию для резервного копирования, если её нет.

Скопировать все файлы с расширением .txt из исходной директории в целевую.

При копировании добавить к имени файла текущую дату в формате ГГГГ-ММ-ДД.

Вывести количество скопированных файлов.

Эталон выполнения

Текст скрипта (код)

Скриншот выполнения с примером

Скриншот результата (файлы с датой в имени)

Критерии оценки

Что проверяется	Балл
Проверка исходной директории	1
Создание целевой директории	1
Копирование только .txt файлов	1
Добавление даты к имени файла	1
Подсчёт и вывод количества файлов	1
Итого	5

Задание 3. Git: локальный и удалённый репозиторий

Выполните следующие действия с Git:

Настроить Git (имя и email пользователя)

Создать локальный репозиторий в папке ~/git-exam
Создать файл hello.txt с текстом «Hello, Git!» и сделать первый коммит
Создать ветку feature, переключиться на неё
Добавить в hello.txt новую строку «Feature branch work» и сделать коммит
Переключиться обратно в ветку main
Создать репозиторий на GitHub
Связать локальный репозиторий с удалённым
Отправить ветку main на GitHub

Эталон выполнения
Отчёт со всеми командами
Скриншот вывода git log --oneline
Скриншот репозитория на GitHub с коммитом
Ссылка на репозиторий (или скриншот)
Критерии оценки

Что проверяется	Балл
Настройка Git выполнена	1
Локальный репозиторий и первый коммит	1
Ветка feature и коммит в ней	1
Удалённый репозиторий создан и связан	1
Код отправлен (push) на GitHub	1
Итого	5

Задание 4. Программа на C: «Угадай число»
Напишите программу на C, которая реализует игру «Угадай число».
Требования:
Программа загадывает случайное число от 1 до 100
Пользователь вводит числа, программа даёт подсказки: «Больше», «Меньше», «Угадал!»
У пользователя не более 10 попыток
После окончания игры программа спрашивает: «Сыграем ещё? y/n»
При вводе y — новая игра, при n — выход

Эталон выполнения
Исходный код программы (файл guess.c)
Скриншот компиляции (gcc guess.c -o guess)
Скриншот выполнения программы (с примером игры)
Критерии оценки

Что проверяется	Балл
Программа компилируется без ошибок	1
Случайное число генерируется	1
Подсказки «Больше»/«Меньше» работают	1
Ограничение на 10 попыток	1
Возможность сыграть ещё	1
Итого	5

Задание 5. Массивы и функции на C
Напишите программу на C, которая работает с массивом целых чисел.
Требования:
Запросить у пользователя размер массива N (не более 20)
Заполнить массив числами, введёнными с клавиатуры
Вывести исходный массив на экран
Найти и вывести минимальный, максимальный элементы и сумму всех элементов (каждое вычисление в отдельной функции)

Отсортировать массив по возрастанию методом «пузырька»
Вывести отсортированный массив
Подсчитать и вывести количество чётных чисел

Эталон выполнения
Исходный код программы (файл array.c)
Скриншот компиляции и выполнения (например, для N=5)
Критерии оценки

Что проверяется	Балл
Ввод N и заполнение массива	1
Функции минимума, максимума, суммы	1
Сортировка «пузырьком»	1
Вывод исходного и отсортированного массивов	1
Подсчёт чётных чисел	1
Итого	5

Задание 6. Работа с файлами на C
Напишите программу file_stats.c, которая анализирует текстовый файл.
Требования:
Программа принимает аргументы командной строки: входной файл и выходной файл
Проверяет, что передано 2 аргумента, иначе выводит инструкцию
Открывает входной файл для чтения, если файл не существует — выводит ошибку
Подсчитывает во входном файле:
количество строк
количество слов (разделители: пробел, табуляция, перевод строки)
количество символов (без учёта перевода строки)
Записывает результаты в выходной файл в формате:
text
Количество строк: X
Количество слов: Y
Количество символов: Z
Закрывает оба файла

Эталон выполнения
Исходный код программы (файл file_stats.c)
Скриншот компиляции
Пример файла input.txt
Скриншот выполнения программы
Полученный файл output.txt (содержимое)
Критерии оценки

Что проверяется	Балл
Проверка аргументов командной строки	1
Проверка существования входного файла	1
Корректный подсчёт строк, слов, символов	1
Запись результатов в выходной файл	1
Программа компилируется и работает	1
Итого	5

Модуль 2. Алгоритмизация и структурное программирование

Задание 1. Строки + аргументы командной строки (C)
Условие
Напишите программу compare.c, которая принимает из командной строки две строки и сравнивает их без учёта регистра, не используя стандартную функцию strcasecmp.

Формат запуска:
bash
./compare "Hello" "HELLO"
Требования:
Проверить, что передано ровно 2 аргумента. Если нет — вывести Usage: ./compare <str1> <str2> и завершиться с кодом 1
Сравнить строки без учёта регистра (своя реализация)
Вывести Equal, если строки равны, иначе Not equal
Завершиться с кодом 0

Эталон выполнения
Исходный код compare.c
Код компилируется без ошибок и предупреждений
Примеры работы:
./compare Hello HELLO → Equal
./compare Hello World → Not equal
./compare Hello → Usage: ./compare <str1> <str2>
Критерии оценки

Что проверяется	Балл
Проверка количества аргументов командной строки	1
Собственная реализация сравнения строк без учёта регистра	2
Корректный вывод результата	1
Код компилируется (gcc -Wall -Wextra -Werror)	1
Итого	5

Задание 2. Структуры (C)
Условие
Определите структуру Student, содержащую:
name — строка (массив char размером 50)
age — целое число
gpa — вещественное число (средний балл)
Напишите программу, которая:
Создаёт массив из 3 студентов (заполнить в коде)
Находит студента с максимальным средним баллом
Выводит его имя и средний балл в формате: Best student: <name>, GPA: <gpa>

Эталон выполнения
Исходный код students.c
Структура определена верно
Массив из 3 студентов создан и заполнен
Функция поиска лучшего реализована
Критерии оценки

Что проверяется	Балл
Структура Student определена верно	1
Массив структур создан и заполнен	1
Поиск студента с максимальным GPA	1
Корректный вывод результата	1
Код компилируется и работает	1
Итого	5

Задание 3. Матрицы (C)
Условие
Определите структуру Matrix, содержащую:

rows — количество строк

cols — количество столбцов

data — указатель на двумерный массив (double**)

Напишите функцию matrix_sum, которая принимает две структуры Matrix и возвращает новую структуру Matrix с суммой матриц.

Требования:

Если размеры матриц не совпадают, вернуть структуру с rows = 0, cols = 0, data = NULL

Память для новой матрицы выделяется динамически

При ошибке выделения памяти — вернуть нулевую матрицу

Эталон выполнения

Исходный код matrix.c (только функция и структура, либо полная программа с примером)

Структура Matrix определена

Проверка совпадения размеров

Динамическое выделение памяти

Критерии оценки

Что проверяется	Балл
Структура Matrix определена верно	1
Проверка совпадения размеров матриц	1
Динамическое выделение памяти под data	1
Корректное сложение всех элементов	1
Обработка ошибок (NULL после malloc)	1
Итого	5

Задание 4. Матрицы + файлы (C)

Условие

Напишите программу transpose.c, которая:

Принимает из командной строки имя входного файла и имя выходного файла

Читает матрицу из входного файла (формат: первая строка — rows cols, затем rows строк по cols чисел)

Вычисляет транспонированную матрицу

Записывает результат в выходной файл в том же формате

Формат запуска:

bash

./transpose input.txt output.txt

Эталон выполнения

Исходный код transpose.c

Проверка аргументов командной строки (если меньше 2 — вывести usage)

Чтение из файла с проверкой ошибок

Динамическое выделение памяти

Транспонирование выполнено верно

Запись в выходной файл в правильном формате

Критерии оценки

Что проверяется	Балл
Проверка аргументов командной строки	1
Корректное чтение из файла	1
Динамическое выделение памяти под матрицу	1
Транспонирование выполнено верно	1
Корректная запись в выходной файл	1
Итого	5

Задание 5. Алгоритмы: сортировка (язык по выбору)

Условие

На любом языке по выбору (C, C++, Java, Kotlin, Swift, Python, Go, C#, JS) реализуйте функцию сортировки массива целых чисел методом «пузырька» (bubble sort) по возрастанию.

Требования:

Функция принимает массив (или список) и его длину

Не использовать встроенные функции сортировки языка

Функция изменяет исходный массив (не создаёт новый)

Эталон выполнения

Исходный код с функцией bubble_sort

Пример использования (в main или тесте)

Результат работы на примере: [64, 34, 25, 12, 22, 11, 90] → [11, 12, 22, 25, 34, 64, 90]

Критерии оценки

Что проверяется	Балл
Выбран корректный язык	1
Реализован именно алгоритм «пузырька»	1
Сортировка по возрастанию	1
Функция изменяет исходный массив	1
Пример работает верно	1
Итого	5

Задание 6. Алгоритмы: рекурсия (язык по выбору)

Условие

На любом языке по выбору (из того же, что и задание 5, или другом) реализуйте рекурсивную функцию для вычисления N-го числа Фибоначчи. Циклы использовать запрещено.

Формула:

fib(0) = 0

fib(1) = 1

fib(n) = fib(n-1) + fib(n-2)

Требования:

Функция принимает целое неотрицательное число n

Возвращает целое число

Обработка n ≤ 0 — возвращать 0

Эталон выполнения

Исходный код с рекурсивной функцией fib

Пример использования (в main или тесте)

Результат для n=10: 55

Критерии оценки

Что проверяется	Балл
Функция рекурсивная (без циклов)	2
Базовые случаи (n=0, n=1) обработаны верно	1
Рекурсивный случай реализован верно	1
Результат для n=10 = 55	1
Итого	5

Модуль 3. Объектно-ориентированное программирование

Задание 1. Класс и инкапсуляция (аналог CPP1 — матрица)

Условие

Создайте класс Matrix, который представляет матрицу вещественных чисел.

Требования:

Закрытые поля: rows (int), cols (int), data (double**)

Публичные методы:

Конструктор по умолчанию (создаёт пустую матрицу 0×0)

Конструктор с параметрами Matrix(int rows, int cols) — создаёт матрицу заданного размера, заполненную нулями

Конструктор копирования

Деструктор (освобождает память)

Геттеры: get_rows(), get_cols()

Метод set_value(int row, int col, double value) — устанавливает значение

Метод get_value(int row, int col) const — возвращает значение

Метод print() const — выводит матрицу на экран

Проверка инкапсуляции: поля rows, cols, data должны быть закрыты (private).

Эталон выполнения

Файлы matrix.hpp и matrix.cpp

Класс Matrix с указанными конструкторами, деструктором и методами

Все методы реализованы корректно

Память выделяется и освобождается правильно (нет утечек)

Критерии оценки

Что проверяется	Балл
Класс определён корректно (поля private)	1
Конструкторы (по умолчанию, с параметрами, копирования)	1
Деструктор освобождает память	1
Геттеры и методы set/get реализованы	1
Метод print выводит матрицу	1
Итого	5

Задание 2. Перегрузка операторов (аналог CPP1)

Условие

Дополните класс Matrix из задания 1 перегрузкой следующих операторов:

operator+ — сложение двух матриц (возвращает новую матрицу)

operator- — вычитание двух матриц

operator* — умножение двух матриц (матричное умножение)

operator== — сравнение матриц (поэлементно)

operator= — присваивание (глубокое копирование)

Требования:

Если размеры матриц не совпадают для +, -, == — выбросить исключение std::invalid_argument

Для умножения — количество столбцов первой должно равняться количеству строк второй, иначе исключение

Оператор = должен корректно работать при самоприсваивании

Эталон выполнения

Файлы matrix.hpp и matrix.cpp с добавленными операторами

Все операторы перегружены

Обработка исключений при некорректных размерах

Оператор присваивания с проверкой на самоприсваивание

Критерии оценки

Что проверяется	Балл
Оператор + перегружен	1
Оператор - перегружен	1
Оператор * перегружен	1
Оператор == перегружен	1

Оператор = перегружен (глубокое копирование)	1
Итого	5

Задание 3. Наследование и полиморфизм

Условие

Создайте иерархию классов для геометрических фигур:

Абстрактный базовый класс Shape с чисто виртуальными методами:

double area() const = 0 — площадь

double perimeter() const = 0 — периметр

void print() const = 0 — вывод информации

Класс Rectangle (прямоугольник), наследник Shape:

Поля: width, height

Конструктор с параметрами

Реализация методов area(), perimeter(), print()

Класс Circle (окружность), наследник Shape:

Поля: radius

Конструктор с параметрами

Реализация методов area(), perimeter(), print()

Напишите функцию print_shape_info, которая принимает указатель на Shape и вызывает его метод print().

Эталон выполнения

Файлы shape.hpp, rectangle.hpp, circle.hpp

Базовый класс с чисто виртуальными методами

Классы-наследники с корректными формулами

Полиморфное поведение (через указатель на базовый класс)

Критерии оценки

Что проверяется	Балл
Абстрактный базовый класс Shape создан	1
Класс Rectangle реализован (наследование, методы)	1
Класс Circle реализован (наследование, методы)	1
Формулы площади и периметра верны	1
Функция print_shape_info демонстрирует полиморфизм	1
Итого	5

Задание 4. Шаблоны (аналог CPP2 — контейнеры)

Условие

Реализуйте шаблонный класс MyVector<T>, аналогичный std::vector.

Требования:

Шаблонный класс с параметром типа T

Закрытые поля: указатель на массив T*, размер (size), вместимость (capacity)

Конструктор по умолчанию

Конструктор с параметром size (создаёт массив заданного размера)

Деструктор

Метод push_back(const T& value) — добавляет элемент в конец (при необходимости увеличивает capacity в 2 раза)

Метод pop_back() — удаляет последний элемент

Метод size() — возвращает текущий размер

Метод at(int index) — доступ по индексу с проверкой границ

Оператор [] — доступ по индексу без проверки границ

Эталон выполнения

Файл my_vector.hpp (шаблонный класс)

Все методы реализованы

Управление памятью корректное (выделение, перевыделение, освобождение)

Шаблон работает с разными типами (int, double, string)

Критерии оценки

Что проверяется	Балл
Шаблонный класс определён	1
Конструкторы и деструктор	1
Метод push_back с увеличение capacity	1
Методы pop_back, size	1
Доступ по индексу (at с проверкой, [] без проверки)	1
Итого	5

Задание 5. Обработка исключений + умные указатели

Условие

Создайте класс SafeArray, который представляет массив фиксированного размера с безопасным доступом.

Требования:

Шаблонный класс SafeArray<T>

Конструктор принимает размер массива

Оператор [] — выбрасывает исключение std::out_of_range при выходе за границы

Метод size() — возвращает размер

Внутри для хранения данных использовать std::unique_ptr<T[]>

В конструкторе, если размер <= 0, выбросить std::invalid_argument

Эталон выполнения

Файл safe_array.hpp (шаблонный класс)

Использование std::unique_ptr для управления памятью

Проверка границ с исключениями

Исключение при некорректном размере

Критерии оценки

Что проверяется	Балл
Шаблонный класс определён	1
Использование std::unique_ptr для данных	1
Проверка границ в операторе [] с исключением	1
Проверка размера в конструкторе	1
Метод size реализован	1
Итого	5

Задание 6. Паттерн проектирования (аналог СРР3/СРР4)

Условие

Реализуйте паттерн «Одиночка» (Singleton) для класса Logger, который занимается записью сообщений в файл.

Требования:

Класс Logger имеет приватный конструктор

Статический метод getInstance() возвращает единственный экземпляр

Метод log(const std::string& message) — записывает сообщение в файл log.txt с временной меткой

Запрещено копирование и присваивание (удалить конструктор копирования и оператор присваивания)

Потокобезопасность не требуется (для упрощения)

Временная метка — текущее время в формате [ГГГГ-ММ-ДД ЧЧ:ММ:СС].

Эталон выполнения
Файлы logger.hpp, logger.cpp
Класс Logger с приватным конструктором
Статический метод getInstance
Метод log с записью в файл
Запрет копирования
Критерии оценки

Что проверяется	Балл
Приватный конструктор	1
Статический метод getInstance (один экземпляр)	1
Метод log записывает в файл с временной меткой	1
Запрещены копирование и присваивание	1
Код компилируется и работает	1
Итого	5

Модуль 4. Современная прикладная разработка

Задание 1. Основы языка
Условие
Напишите программу, которая:
Принимает из командной строки или стандартного ввода целое число N
Создаёт массив (или список) из N случайных целых чисел в диапазоне от 1 до 100
Выводит массив на экран
Находит и выводит:
Минимальный элемент
Максимальный элемент
Сумму всех элементов
Среднее арифметическое
Требования: использовать стандартные конструкции языка (циклы, массивы/списки, функции).
Пример работы (для Python)
text
Введите N: 5
Массив: [23, 45, 12, 78, 34]
Минимум: 12
Максимум: 78
Сумма: 192
Среднее: 38.4

Эталон выполнения
Исходный код программы
Программа корректно обрабатывает ввод
Массив заполняется случайными числами
Все вычисления выполняются без использования встроенных агрегирующих функций (min, max, sum разрешены, но если их нет — плюс)
Вывод соответствует формату
Критерии оценки

Что проверяется	Балл
Корректный ввод N	1
Заполнение массива случайными числами	1
Нахождение минимума и максимума	1
Нахождение суммы и среднего	1
Корректный вывод	1
Итого	5

Задание 2. Объектно-ориентированное программирование

Условие

Создайте класс BankAccount, представляющий банковский счёт.

Требования:

Поля:

accountNumber (строка, генерируется автоматически при создании)

ownerName (строка, задаётся при создании)

balance (вещественное число, по умолчанию 0)

Методы:

deposit(amount) — пополнение счёта (сумма должна быть положительной)

withdraw(amount) — снятие со счёта (нельзя снять больше, чем есть на счёте)

getBalance() — возвращает текущий баланс

getAccountInfo() — возвращает строку с информацией о счёте

При попытке снять больше доступной суммы — выбросить исключение или вернуть ошибку.

Эталон выполнения

Исходный код класса

Демонстрация работы в main или тестах

Инкапсуляция соблюдена (поля приватные)

При попытке снять больше — ошибка обработана

Критерии оценки

Что проверяется	Балл
Класс создан, поля приватные	1
Конструктор с параметром ownerName	1
Метод deposit работает корректно	1
Метод withdraw работает с проверкой	1
Методы getBalance и getAccountInfo	1
Итого	5

Задание 3. Функциональное программирование

Условие

Напишите программу, которая:

Создаёт список строк: ["apple", "banana", "cherry", "date", "elderberry", "fig", "grape"]

С помощью функций высшего порядка (map/filter/reduce или аналоги):

Отфильтровывает строки длиной менее 5 символов

Преобразует оставшиеся строки в верхний регистр

Сортирует их по алфавиту

Объединяет в одну строку через запятую

Запрещено использовать явные циклы (for, while). Только функции высшего порядка и лямбда-выражения.

Пример результата

text

BANANA, CHERRY, ELDERBERRY, GRAPE

Эталон выполнения

Исходный код программы

Используются функции высшего порядка (map, filter, reduce / LINQ / stream / list comprehensions с условием и т.д.)

Нет явных циклов

Результат соответствует ожидаемому

Критерии оценки

Что проверяется	Балл
Фильтрация строк (длина ≥ 5)	1
Преобразование в верхний регистр	1
Сортировка по алфавиту	1
Объединение через запятую	1
Отсутствие явных циклов	1
Итого	5

Задание 4. Консольная игра (Rogue-like) с curses

Условие

Разработайте консольную игру «Лабиринт» с использованием библиотеки curses (или её аналога для выбранного языка: dotnet-curses для C#, JCurses для Java, blessed для JS, curses для Python, cinterop для Kotlin, vapor для Swift, go-curses для Go).

Игровая механика:

Игровое поле — лабиринт 10×10

Стены обозначаются #, проходы — пробел

Игрок обозначается @, выход — X

Управление стрелками (вверх, вниз, влево, вправо)

При достижении выхода — победа, выход из игры

При попытке врезаться в стену — игрок не двигается

Требования:

Использовать curses для отрисовки

Неблокирующий ввод (игра реагирует на клавиши в реальном времени)

Эталон выполнения

Исходный код игры

Корректная инициализация curses

Отображение лабиринта, игрока, выхода

Управление стрелками

Сообщение о победе при достижении выхода

Критерии оценки

Что проверяется	Балл
Лабиринт отображается корректно	1
Игрок и выход отображаются	1
Управление стрелками работает	1
Обработка столкновений со стенами	1
Победа при достижении выхода	1
Итого	5

Задание 5. Веб-приложение (CRUD + база данных)

Условие

Создайте веб-приложение с использованием выбранного фреймворка:

Язык Фреймворк

C# ASP.NET Core

Java Spring Boot

JavaScript Nest.js

Kotlin Ktor

Python Flask

Swift Vapor

Go Gin / Echo / net/http

Функциональность:

Модель Task (id, title, description, completed, created_at)

REST API эндпоинты:

GET /tasks — получить все задачи

GET /tasks/{id} — получить задачу по id

POST /tasks — создать новую задачу
PUT /tasks/{id} — обновить задачу
DELETE /tasks/{id} — удалить задачу
Данные хранятся в базе данных (любой СУБД на выбор: SQLite, PostgreSQL, MySQL)
Использовать ORM (Entity Framework, Spring Data JPA, TypeORM, Exposed, SQLAlchemy, Fluent, GORM)

Эталон выполнения
Исходный код приложения
Конфигурация подключения к БД
Миграции (если используются)
Все CRUD-эндпоинты реализованы и работают
Критерии оценки

Что проверяется	Балл
Модель Task создана	1
GET-эндпоинты работают	1
POST-эндпоинт работает	1
PUT-эндпоинт работает	1
DELETE-эндпоинт работает	1
Итого	5

Задание 6. JWT-авторизация + BrickGame v3.0 (обратная совместимость)
Условие
Разработайте веб-сервер (API) для игры BrickGame, который поддерживает:
JWT-авторизацию:
Регистрация пользователя (POST /register)
Вход (POST /login) — возвращает JWT-токен
Защищённые эндпоинты требуют валидный JWT
Игровую логику BrickGame (тетрис-лайк) на сервере или клиенте (по выбору):
Хранение рекордов пользователей в БД
Получение рекорда (GET /record — защищённый)
Сохранение нового рекорда (POST /record — защищённый)
Обратную совместимость:
API имеет две версии: v1 (старая, без JWT) и v2 (новая, с JWT)
Клиенты v1 продолжают работать без авторизации
Клиенты v2 используют JWT

Эталон выполнения
Исходный код сервера
Эндпоинты /register, /login с JWT
Защищённые эндпоинты с проверкой токена
Две версии API (v1 и v2)
Сохранение и получение рекордов из БД
Критерии оценки

Что проверяется	Балл
Регистрация и вход с JWT	1
Защищённые эндпоинты работают	1
Две версии API (обратная совместимость)	1
Сохранение рекордов в БД	1
Получение рекордов из БД	1
Итого	5

Модуль 5. Инфраструктурные решения и управление данными

Задание 1. Установка и настройка Linux

Условие

Установите Linux (Ubuntu 22.04 / аналогичный дистрибутив) на виртуальную машину и выполните базовую настройку.

Требования:

Установить систему с минимальной конфигурацией (без графического окружения или с базовым)

Создать пользователя student с паролем

Настроить sudo для пользователя student без запроса пароля

Обновить систему через пакетный менеджер (apt update && apt upgrade)

Установить пакеты: git, curl, htop, nano (или vim)

Настроить SSH-сервер: установить, запустить, добавить свой публичный ключ для доступа без пароля

Эталон выполнения

Скриншот терминала с выводом uname -a

Скриншот с выводом sudo -l (показывает права без пароля)

Скриншот с подключением по SSH через ключ

Файл /etc/hostname (содержимое)

Чек-лист со всеми выполненными пунктами

Критерии оценки

Что проверяется	Балл
Система установлена	1
Пользователь student создан, sudo настроен	1
Система обновлена, пакеты установлены	1
SSH-сервер настроен, ключ добавлен	1
Оформление отчёта (все скриншоты и файлы)	1
Итого	5

Задание 2. Настройка сети в Linux

Условие

На виртуальной машине с Linux выполните настройку сети.

Требования:

Определить и записать в отчёт:

IP-адрес виртуальной машины

MAC-адрес сетевого интерфейса

Шлюз по умолчанию

DNS-серверы

Пропинговать 3 внешних хоста (8.8.8.8, ya.ru, любой свой)

Настроить статический IP вместо DHCP (в конфигурации Netplan или interfaces)

Добавить в /etc/hosts запись: 127.0.0.1 myhost.local

Проверить маршрут до ya.ru с помощью traceroute (или tracepath)

Эталон выполнения

Вывод ip a и ip r (скриншот или текст)

Вывод ping -c 4 ya.ru

Файл конфигурации сети (Netplan или interfaces) после настройки статического IP

Вывод traceroute ya.ru (первые 10 хопов)

Пояснение ключевых параметров (IP, маска, шлюз, DNS)

Критерии оценки

Что проверяется	Балл
Определены сетевые параметры	1
Ping до внешних хостов работает	1

Статический IP настроен	1
Запись в /etc/hosts добавлена	1
traceroute выполнен	1
Итого	5

Задание 3. Docker и CI/CD

Условие

Разработайте простой CI/CD-пайплайн для одного из проектов (например, SimpleBashUtils или любой другой утилиты на C).

Требования:

Написать Dockerfile, который:

Создаёт образ на основе ubuntu:latest

Устанавливает gcc, make, git

Копирует исходный код проекта

Выполняет сборку (make)

Запускает тесты

Написать конфигурацию CI/CD (GitLab CI или GitHub Actions):

build — сборка Docker-образа

test — запуск тестов внутри контейнера

Загрузить образ в реестр (Docker Hub / GitHub Container Registry)

В отчёте описать этапы пайплайна и приложить скриншоты

Эталон выполнения

Файл Dockerfile

Файл .gitlab-ci.yml или .github/workflows/ci.yml

Скриншот успешного выполнения пайплайна

Ссылка на образ в реестре (или скриншот)

Инструкция по сборке и запуску

Критерии оценки

Что проверяется	Балл
Dockerfile корректен, образ собирается	1
CI/CD конфигурация написана	1
Пайплайн проходит успешно (build + test)	1
Образ загружен в реестр	1
Отчёт с описанием и скриншотами	1
Итого	5

Задание 4. SQL: создание таблиц и вставка данных

Условие

Работа с базой данных (PostgreSQL / SQLite / MySQL).

Часть 1. Создание таблиц

Создайте таблицы для учёта студентов и их оценок:

sql

-- Таблица students

-- id (PRIMARY KEY), name (TEXT), email (TEXT UNIQUE), enrolled_at (DATE)

-- Таблица courses

-- id (PRIMARY KEY), title (TEXT), hours (INTEGER)

-- Таблица grades

-- id (PRIMARY KEY), student_id (FOREIGN KEY → students.id),

-- course_id (FOREIGN KEY → courses.id), grade (INTEGER), date (DATE)

Часть 2. Вставка данных

Вставьте:

3 студентов

2 курса
5 оценок (у каждого студента хотя бы по одной)

Эталон выполнения
SQL-скрипт создания таблиц (CREATE TABLE)
SQL-скрипт вставки данных (INSERT)
Скриншот результата SELECT * FROM students;, SELECT * FROM courses;, SELECT * FROM grades;
Критерии оценки

Что проверяется	Балл
Таблица students создана с ограничениями	1
Таблица courses создана	1
Таблица grades создана с внешними ключами	1
Вставлены 3+ студента, 2+ курса, 5+ оценок	1
Все данные корректны (внешние ключи ссылаются на существующие записи)	1
Итого	5

Задание 5. SQL: выборка, фильтрация, группировка, JOIN
Условие
На основе таблиц из задания 4 напишите SQL-запросы:
Вывести всех студентов, зачисленных после 2024-01-01 (если дат нет — можно добавить)
Вывести список курсов с количеством часов больше 30
Вывести среднюю оценку каждого студента (имя + средний балл)
Вывести список студентов, у которых есть хотя бы одна оценка ниже 3 (или ниже 60, в зависимости от шкалы)
Вывести для каждого курса: название курса, максимальную оценку, минимальную оценку, количество студентов
Требования: использовать WHERE, GROUP BY, AVG, MIN, MAX, COUNT, JOIN.

Эталон выполнения
5 SQL-запросов (текст)
Скриншоты результатов выполнения каждого запроса
Критерии оценки

Что проверяется	Балл
Запрос 1 (фильтрация по дате)	1
Запрос 2 (фильтрация по часам)	1
Запрос 3 (средняя оценка студента)	1
Запрос 4 (оценки ниже порога)	1
Запрос 5 (агрегации по курсам)	1
Итого	5

Задание 6. SQL: обновление и удаление данных
Условие
На основе таблиц из задания 4 выполните операции модификации данных:
UPDATE: увеличить количество часов всех курсов на 10%
UPDATE: студенту с id = 1 изменить email на newemail@example.com
DELETE: удалить все оценки ниже 3 (или ниже 60)
DELETE: удалить студента с id = 3 (предварительно удалив его оценки, используя ON DELETE CASCADE или отдельный DELETE)
SELECT после каждого изменения, чтобы убедиться в корректности
Требования: показать SQL-запросы и результаты до/после.

Эталон выполнения

SQL-запросы на UPDATE и DELETE (текст)

Скриншоты SELECT до и после каждого изменения

Критерии оценки

Что проверяется	Балл
UPDATE часов курсов (+10%)	1
UPDATE email студента	1
DELETE низких оценок	1
DELETE студента (с каскадом или предварительным удалением оценок)	1
Результаты подтверждены SELECT	1
Итого	5

Модуль 6. Старт бизнеса

Задание 1. Что такое Lean Startup

Условие

Выберите любую бизнес-идею (реальную или учебную). Опишите её в 2–3 предложениях.

Требования:

Сформулируйте гипотезу ценности (предположение о том, какую проблему решает продукт и для кого)

Сформулируйте гипотезу роста (предположение о том, как продукт будет привлекать новых пользователей)

Опишите MVP (минимально жизнеспособный продукт) — какие минимальные функции нужны для проверки гипотез

Постройте цикл Build → Measure → Learn для этого MVP (что делаем → как измеряем → что узнаём)

Эталон выполнения

Текстовый документ (или слайд) с описанием бизнес-идеи

Две гипотезы (ценность и рост)

Описание MVP (3–5 пунктов, что входит)

Схема или текст цикла Build → Measure → Learn

Критерии оценки

Что проверяется	Балл
Бизнес-идея описана чётко	1
Гипотеза ценности сформулирована корректно	1
Гипотеза роста сформулирована корректно	1
MVP описан (минимальный набор функций)	1
Цикл Build → Measure → Learn проработан	1
Итого	5

Задание 2. Как быстро тестировать гипотезы с помощью Custdev

Условие

Для той же бизнес-идеи (из задания 1) разработайте план Custdev-интервью.

Требования:

Сформулируйте 3 гипотезы, которые нужно проверить через Custdev (например, о проблеме, о решении, о цене)

Напишите скрипт интервью (10–15 вопросов). Вопросы должны быть открытыми, не наводящими, без продажи

Опишите портрет респондента (кто целевая аудитория, где её искать, критерии отбора)

Укажите, сколько интервью нужно провести для проверки гипотез (минимум 5–10)

Опишите, как будете анализировать результаты (выявление паттернов,

подтверждение/опровержение гипотез)

Эталон выполнения

Текстовый документ с тремя гипотезами

Скрипт интервью (10–15 вопросов)

Портрет респондента (3–5 критериев)

План анализа результатов

Критерии оценки

Что проверяется	Балл
Три гипотезы сформулированы (проблема, решение, цена)	1
Скрипт интервью готов (вопросы открытые, ненаводящие)	1
Портрет респондента описан	1
План по количеству интервью указан	1
Метод анализа результатов описан	1
Итого	5

Задание 3. Как оценивать рынок и избежать типичных ошибок

Условие

Для той же бизнес-идеи (из задания 1) проведите оценку рынка.

Требования:

Рассчитайте TAM (Total Addressable Market) — общий объём рынка для вашего продукта

Рассчитайте SAM (Serviceable Available Market) — доступный сегмент рынка

Рассчитайте SOM (Serviceable Obtainable Market) — реально достижимая доля (например, 5–10% от SAM)

Укажите источники данных для расчётов (статистика, отчёты, аналитика)

Опишите 3 типичные ошибки при оценке рынка (и как вы их избежали в своём расчёте)

Эталон выполнения

Таблица с цифрами TAM, SAM, SOM (в деньгах или единицах)

Источники данных (ссылки или названия отчётов)

Список 3 ошибок с пояснением, как избежали

Критерии оценки

Что проверяется	Балл
TAM рассчитан (с обоснованием)	1
SAM рассчитан (с обоснованием)	1
SOM рассчитан (с обоснованием)	1
Источники данных указаны	1
3 типичные ошибки описаны	1
Итого	5

Задание 4. Как масштабировать бизнес за пределы региона

Условие

Для той же бизнес-идеи (из задания 1) разработайте стратегию масштабирования.

Требования:

Выберите один регион для первой экспансии (соседний город, страну или международный рынок)

Обоснуйте выбор региона по 5 критериям (размер рынка, близость, язык/культура, конкуренция, регуляторика)

Опишите адаптацию продукта под новый регион (язык, функциональность, упаковка, цена)

Составьте roadmap масштабирования по шагам с указанием сроков (первые 3–6 месяцев)

Опишите риски масштабирования (минимум 3) и способы их снижения

Эталон выполнения

Название выбранного региона

Обоснование по 5 критериям (таблица или текст)

Список адаптаций продукта

Дорожная карта (пошагово, с датами)

Риски и меры снижения

Критерии оценки

Что проверяется	Балл
Регион выбран и обоснован	1
5 критериев оценки региона применены	1
Адаптация продукта описана	1
Roadmap составлен	1
Риски и снижения описаны	1
Итого	5

Задание 5. Как начать бизнес с франшизой

Условие

Проведите анализ франшизы (выберите реальную или гипотетическую).

Требования:

Выберите 3 франшизы из интересующей ниши (можно реальные из открытых каталогов)

Составьте сравнительную таблицу по параметрам:

Паушальный взнос

Роялти (процент или фиксированная сумма)

Инвестиции (стартовые)

Срок окупаемости (по данным франчайзора)

Обучение и поддержка (есть/нет, что входит)

Рассчитайте примерную прибыль и срок окупаемости для одной из франшиз

Составьте чек-лист из 10 вопросов для проверки франчайзора (что спросить перед покупкой)

Сделайте вывод: в каком случае франшиза выгоднее открытия бизнеса «с нуля» (2–3 аргумента)

Эталон выполнения

Сравнительная таблица на 3 франшизы

Расчёт окупаемости (по одной франшизе)

Чек-лист из 10 вопросов

Вывод (2–3 аргумента)

Критерии оценки

Что проверяется	Балл
3 франшизы выбраны, таблица составлена	1
Расчёт окупаемости выполнен	1
Чек-лист из 10 вопросов готов	1
Вывод обоснован (2–3 аргумента)	1
Все разделы оформлены аккуратно	1
Итого	5

Модуль 7. Рост бизнеса

Задание 1. Продуктовый подход и работа с гипотезами

Условие

Для вашего бизнес-проекта (или выбранного учебного проекта) выполните продуктовый анализ.

Требования:

Сформулируйте 5–7 гипотез по структуре: «Если мы сделаем X, то получим Y, потому что Z»

Проведите приоритизацию гипотез методом ICE (Impact, Confidence, Ease) или RICE (Reach, Impact, Confidence, Effort) — заполните таблицу с баллами

Выберите топ-3 гипотезы для проверки

Для одной гипотезы (на выбор) опишите эксперимент: что делаем, как измеряем, какой результат считаем успехом (метрики)

Назовите 3 типа метрик, которые нужно отслеживать при проверке гипотез (например, качественные, количественные, продуктовые)

Эталон выполнения

Список из 5–7 гипотез

Таблица приоритизации (ICE или RICE) с баллами

Топ-3 гипотезы

Описание эксперимента для одной гипотезы (шаги, метрики успеха)

3 типа метрик с пояснением

Критерии оценки

Что проверяется	Балл
5–7 гипотез сформулированы	1
Приоритизация выполнена (ICE/RICE)	1
Топ-3 гипотезы выбраны обоснованно	1
Эксперимент описан (что, как, метрики успеха)	1
3 типа метрик названы	1
Итого	5

Задание 2. Custdev и концепция Jobs to be Done (JTBD)

Условие

Для вашего бизнес-проекта примените концепцию JTBD.

Требования:

Опишите 3 «работы» (jobs) , которые клиент «нанимает» ваш продукт выполнить (функциональные, эмоциональные, социальные)

Для каждой работы опишите: ситуацию (контекст), мотивацию (почему это важно), ожидаемый результат, альтернативы (как клиент решает проблему сейчас)

Напишите скрипт JTBD-интервью (5–7 вопросов), направленный на выявление «работ»

Объясните, чем JTBD отличается от Custdev (традиционного подхода) — 2–3 ключевых отличия

Укажите, как результаты JTBD-анализа повлияют на ваш продукт (что измените)

Эталон выполнения

3 работы (job) с описанием ситуации, мотивации, результата, альтернатив

Скрипт JTBD-интервью

Таблица или текст отличий JTBD от Custdev

План изменений в продукте

Критерии оценки

Что проверяется	Балл
3 работы описаны полно	1
По каждой работе — ситуация, мотивация, результат, альтернативы	1
Скрипт JTBD-интервью готов	1

Отличия JTBD от Custdev объяснены	1
Влияние на продукт описано	1
Итого	5

Задание 3. Анализ рынка

Условие

Проведите углублённый анализ рынка для вашего бизнес-проекта.

Требования:

Проведите PESTEL-анализ (Политические, Экономические, Социальные, Технологические, Экологические, Юридические факторы) — минимум по 2 фактора на категорию

Проведите анализ 5 сил Портера (угроза новых игроков, заменители, власть поставщиков, власть покупателей, интенсивность конкуренции)

Постройте карту конкурентов (две оси: например, «цена» и «качество/функциональность»). Разместите 5–10 конкурентов + свой продукт

Выделите 3 конкурентных преимущества вашего продукта (чем отличаетесь)

Сделайте SWOT-анализ (сильные и слабые стороны, возможности и угрозы)

Эталон выполнения

Таблица PESTEL (по 2+ фактора на категорию)

Таблица или текст по 5 силам Портера

Карта конкурентов (нарисованная или табличная)

3 конкурентных преимущества

SWOT-матрица

Критерии оценки

Что проверяется	Балл
PESTEL-анализ выполнен	1
5 сил Портера проанализированы	1
Карта конкурентов построена	1
3 конкурентных преимущества выделены	1
SWOT-анализ выполнен	1
Итого	5

Задание 4. Продажи через ценность (Value Selling)

Условие

Разработайте стратегию продаж через ценность для вашего бизнес-проекта.

Требования:

Заполните Value Proposition Canvas (канву ценностного предложения):

Профиль клиента: боли, выгоды, работы

Карта ценности: продукты/услуги, облегчение болей, создание выгод

Сформулируйте ценностное предложение (1–2 предложения) по шаблону: «Мы помогаем [клиент] [решить проблему] с помощью [уникальное решение], в отличие от [альтернатива]»

Напишите скрипт SPIN-вопросов (ситуация, проблема, извлечение, ценность) — 5–7 вопросов для разговора с клиентом

Подготовьте 3 возражения клиентов и ответы на них (с переводом из цены в ценность)

Напишите питч (1–2 минуты, текст) для презентации ценности вашего продукта

Эталон выполнения

Заполненная канва Value Proposition Canvas (таблица или схема)

Ценностное предложение (1–2 предложения)

Скрипт SPIN-вопросов (5–7)

3 возражения и ответы

Питч (текст)

Критерии оценки

Что проверяется	Балл
Value Proposition Canvas заполнена	1
Ценностное предложение сформулировано	1
SPIN-скрипт готов	1
3 возражения и ответы проработаны	1
Питч готов	1
Итого	5

Задание 5. Финансовые модели и юнит-экономика

Условие

Постройте финансовую модель и рассчитайте юнит-экономику для вашего бизнес-проекта.

Требования:

Рассчитайте CAC (Customer Acquisition Cost) — стоимость привлечения одного клиента (укажите допущения)

Рассчитайте LTV (Lifetime Value) — сколько денег приносит один клиент за всё время

Рассчитайте ARPU (Average Revenue Per User) — средняя выручка на одного клиента

Вычислите соотношение LTV / CAC (должно быть > 3 для здорового бизнеса)

Постройте финансовую модель на 12 месяцев (по месяцам):

Доходы (по источникам)

Расходы (постоянные и переменные)

Прибыль / убыток

Денежный поток (cash flow)

Точка безубыточности (break-even point)

Допущения: можно использовать реалистичные цифры (или учебные). Главное — логика расчёта.

Эталон выполнения

Расчёт CAC с допущениями

Расчёт LTV с допущениями

Расчёт ARPU

Соотношение LTV / CAC

Финансовая модель на 12 месяцев (таблица Excel / Google Sheets или в текстовом виде)

Критерии оценки

Что проверяется	Балл
CAC рассчитан (с допущениями)	1
LTV рассчитан (с допущениями)	1
ARPU рассчитан	1
LTV / CAC рассчитан и прокомментирован	1
Финансовая модель на 12 месяцев построена	1
Итого	5

Модуль 8. Продуктовый подход в бизнесе

Задание 1. Личное видение. Устойчивая мотивация основателя

Условие

Сформулируйте личное видение как основателя и проанализируйте свою мотивацию.

Требования:

Напишите личное видение (1–2 абзаца) на 5–10 лет: что вы создаёте, для кого, как меняете мир, какими будете через 5–10 лет

Сформулируйте миссию (одно предложение) — зачем существует ваш проект

Определите 3 источника вашей мотивации (внутренние: смысл, автономия, мастерство, признание, влияние)

Опишите 3 признака выгорания и 3 способа профилактики (личных, не общих)

Составьте список ритуалов (ежедневных или еженедельных) для поддержания энергии и фокуса

Эталон выполнения

Текст личного видения

Миссия (одно предложение)

3 источника мотивации

3 признака выгорания + 3 способа профилактики

Список ритуалов

Критерии оценки

Что проверяется	Балл
Личное видение сформулировано	1
Миссия сформулирована	1
3 источника мотивации названы	1
Выгорание: признаки + профилактика	1
Ритуалы описаны	1
Итого	5

Задание 2. Технологическое предпринимательство. Основные понятия

Условие

Проведите анализ вашего проекта (или учебного) через призму технологического предпринимательства.

Требования:

Дайте определение технологического стартапа (своими словами) и отличие от традиционного малого бизнеса (3 отличия)

Определите тип инновации вашего продукта (продуктовая, процессная, маркетинговая, организационная) — объясните почему

Назовите технологическую платформу вашего продукта (свою или партнёрскую) — например, iOS, Android, AWS, Telegram Bot API

Опишите жизненный цикл технологического стартапа (5 этапов: pre-seed → seed → early → growth → exit) с примером для своего проекта

Перечислите 3 способа защиты интеллектуальной собственности (патент, ноу-хау, коммерческая тайна) и укажите, какой подходит для вашего проекта

Эталон выполнения

Определение технологического стартапа + 3 отличия

Тип инновации с обоснованием

Технологическая платформа

Жизненный цикл стартапа с примерами

3 способа защиты ИС

Критерии оценки

Что проверяется	Балл
Определение и отличия	1
Тип инновации определён	1
Технологическая платформа названа	1
Жизненный цикл описан	1

Защита ИС описана	1
Итого	5

Задание 3. Стратегия подрыва отраслей с помощью технологий

Условие

Проанализируйте стратегию подрыва для вашего проекта.

Требования:

Объясните концепцию подрывных инноваций (Кристенсен): поддерживающие vs подрывные (2–3 предложения)

Назовите 3 примера известных подрывных инноваций (Netflix, Uber, Airbnb, Zoom и др.) — кратко, по 1–2 предложения на пример

Выберите лидера в вашей отрасли (конкретную компанию) и опишите 3 его «слепые зоны» (что он игнорирует: сегмент, функцию, цену, бизнес-модель)

Сформулируйте гипотезу подрыва для вашего проекта: с какого сегмента заходите (low-end или new-market), почему лидеры не ответят

Нарисуйте график «эволюция технологии vs требования рынка» (качественно: линия требований рынка, линия производительности продукта лидера, линия вашего стартапа)

Эталон выполнения

Объяснение подрывных инноваций

3 примера

Лидер в отрасли + 3 слепые зоны

Гипотеза подрыва

График (рисунок или текстовое описание)

Критерии оценки

Что проверяется	Балл
Концепция объяснена	1
3 примера названы	1
Лидер + 3 слепые зоны	1
Гипотеза подрыва сформулирована	1
График построен	1
Итого	5

Задание 4. Подрывные инновации или как стартапу победить лидера рынка

Условие

Разработайте конкретную стратегию, как стартап может победить лидера рынка.

Требования:

Выберите стратегию подрыва (low-end disruption — дешёвый сегмент, или new-market disruption — новый рынок) и обоснуйте

Опишите 3 этапа подрыва для вашего случая: вход → скрытый рост → переключение → вытеснение

Назовите 3 барьера для лидера (почему он не сможет ответить быстро) — например, бизнес-модель, каналы продаж, культура, KPI

Опишите точку входа: недопотреблённые клиенты / переобслуженные клиенты / новые сценарии

Напишите асимметричную войну: в чём ваше преимущество, которое лидер не может скопировать (2–3 пункта)

Эталон выполнения

Выбранный тип подрыва с обоснованием

4 этапа подрыва (описание)

3 барьера для лидера

Точка входа

Асимметричное преимущество (2–3 пункта)

Критерии оценки

Что проверяется	Балл
Тип подрыва выбран и обоснован	1
Этапы подрыва описаны	1
3 барьера для лидера названы	1
Точка входа определена	1
Асимметричное преимущество описано	1
Итого	5

Задание 5. Стратегия голубого океана

Условие

Примените стратегию голубого океана к вашему проекту.

Требования:

Объясните концепцию голубого океана (Ким и Моборн) vs красного океана (2–3 предложения)

Постройте стратегическую канву (график кривой ценности) для вашей отрасли:

Ось X: 5–7 факторов конкуренции (цена, качество, удобство, сервис, ассортимент, скорость, дизайн)

Ось Y: уровень предложения (низкий → высокий)

Нанесите кривые: лидеры отрасли (1–2), ваш продукт (текущий), ваш продукт (голубой океан)

Заполните матрицу «убрать — уменьшить — повысить — создать» (минимум 3 пункта в каждой категории)

Сформулируйте новую кривую ценности (чем вы отличаетесь от конкурентов)

Назовите один из 6 путей создания голубого океана, который вы использовали

Эталон выполнения

Объяснение стратегии голубого океана

Стратегическая канва (таблица или описание графика)

Матрица «убрать-уменьшить-повысить-создать»

Новая кривая ценности

Путь создания голубого океана

Критерии оценки

Что проверяется	Балл
Концепция объяснена	1
Стратегическая канва построена	1
Матрица заполнена (3+ в каждой)	1
Новая кривая ценности сформулирована	1
Путь создания указан	1
Итого	5

Задание 6. Факторы успеха стартапа. Команда основателей

Условие

Проведите анализ факторов успеха вашего стартапа и спроектируйте команду основателей.

Требования:

Назовите топ-5 причин провала стартапов по исследованиям CB Insights (списком)

Опишите идеальный профиль команды из 3 основателей для вашего проекта (роли: бизнес, технологии, продукт — или «хакер, бизнесмен, дизайнер»)

Рассчитайте vesting (4 года + cliff 1 год) для 3 сооснователей с разными долями (например, 40%, 30%, 30%)

Напишите 5 культурных ценностей вашего стартапа (по 1–2 предложения на каждую)

Составьте профиль первого сотрудника (кого наймёте первым, какие навыки, почему именно его)

Эталон выполнения
Топ-5 причин провала
Идеальный профиль команды (3 роли)
Расчёт vesting
5 культурных ценностей
Профиль первого сотрудника
Критерии оценки

Что проверяется	Балл
5 причин провала названы	1
Профиль команды описан	1
Vesting рассчитан	1
5 культурных ценностей сформулированы	1
Профиль первого сотрудника описан	1
Итого	5

Задание 7. Клиент и его потребности. Востребованный продукт
Условие
Проведите анализ соответствия вашего продукта рынку (Product-Market Fit).
Требования:
Объясните понятие Product-Market Fit (PMF) и перечислите 3 признака достижения PMF (органический рост, retention, сарафанное радио)
Постройте Customer Journey Map (CJM) для вашего продукта: 5–7 этапов от осознания проблемы до лояльности, для каждого этапа укажите действия клиента, боли и возможности
Заполните Kano-модель для вашего продукта: по 3 потребности каждого типа (базовые, линейные, восторгающие)
Проведите опрос на PMF (вопрос: «Как сильно вы расстроитесь, если продукт исчезнет?») — представьте гипотетические результаты (например, 45% сказали «очень расстроюсь»). Сделайте вывод о готовности к PMF
Напишите план из 5 итераций по достижению PMF (что будете менять в продукте и как проверять)

Эталон выполнения
Объяснение PMF, 3 признака
CJM (таблица или схема)
Kano-модель (9 потребностей: 3+3+3)
Опрос PMF с гипотетическими результатами и выводом
План из 5 итераций
Критерии оценки

Что проверяется	Балл
PMF объяснён, 3 признака названы	1
CJM построена (5–7 этапов)	1
Kano-модель заполнена (9 потребностей)	1
Опрос PMF и вывод	1
План из 5 итераций	1
Итого	5

Модуль 9. Азбука стартапа

Задание 1. Выбор организационно-правовой формы и расчёт налоговой нагрузки
Условие

Для вашего бизнес-проекта выберите организационно-правовую форму и рассчитайте налоговую нагрузку.

Требования:

Сравните 3 организационно-правовые формы (например, ИП, ООО, самозанятость) по 5 критериям:

Сложность регистрации

Ответственность (полная / в пределах уставного капитала)

Возможность вывести прибыль

Статус для контрагентов (корпоративный или нет)

Стоимость регистрации и обслуживания

Выберите оптимальную форму для вашего проекта и обоснуйте выбор

Сравните 3 налоговых режима (например, УСН 6% (доходы), УСН 15% (доходы минус расходы), НПД (самозанятость), ОСНО) по 4 критериям:

Ставка налога

Объект налогообложения

Ограничения (по доходам, численности сотрудников)

Сложность отчётности

Рассчитайте налоговую нагрузку для выбранного режима на примере:

Доходы за месяц: 1 000 000 руб.

Расходы (аренда, зарплата, материалы): 600 000 руб.

Покажите расчёт налога (цифры)

Сделайте вывод: какая форма и режим подходят для стартапа на ранней стадии

Эталон выполнения

Сравнительная таблица 3 ОПФ

Обоснование выбора

Сравнительная таблица 3 налоговых режимов

Расчёт налоговой нагрузки (с цифрами)

Вывод по стартапу

Критерии оценки

Что проверяется	Балл
Сравнение ОПФ (3 формы, 5 критериев)	1
Выбор и обоснование ОПФ	1
Сравнение налоговых режимов (3 режима, 4 критерия)	1
Расчёт налоговой нагрузки (цифры)	1
Вывод	1
Итого	5

Задание 2. Налоговая модель и управляемость бизнеса с первого дня

Условие

Разработайте налоговую модель для вашего стартапа с учётом управляемости бизнеса.

Требования:

Опишите налоговую модель вашего бизнеса:

Какие налоги платите (НДС, налог на прибыль/доходы, страховые взносы, НДФЛ)

Периодичность уплаты (ежемесячно, ежеквартально)

Кто отвечает за расчёт и уплату (вы сами, бухгалтер, аутсорсинг)

Опишите систему учёта с первого дня:

Как фиксируете доходы и расходы (таблица Excel, облачный сервис, бухгалтерская программа)

Как работаете с первичными документами (договоры, акты, счета, чеки)

Как храните документы (бумажные / электронные)

Составьте график налоговых платежей на первый квартал (по месяцам, с суммами — учебными)

Опишите 3 риска налоговых ошибок на старте и способы их предотвращения

Назовите 3 онлайн-сервиса для бухгалтерии стартапа (например, Тинькофф Бизнес, Мой налог, Контур.Эльба, 1С:Фреш)

Эталон выполнения

Налоговая модель (какие налоги, периодичность, ответственность)

Система учёта с первого дня

График налоговых платежей (3 месяца)

3 риска и их предотвращение

3 онлайн-сервиса

Критерии оценки

Что проверяется	Балл
Налоговая модель описана	1
Система учёта описана	1
График платежей составлен	1
3 риска и предотвращение	1
3 онлайн-сервиса названы	1
Итого	5

Задание 3. Финансовое управление бизнесом

Условие

Разработайте систему финансового управления для вашего стартапа.

Требования:

Составьте прогноз движения денежных средств (Cash Flow) на 6 месяцев (по месяцам):

Поступления от клиентов

Операционные расходы (аренда, зарплата, реклама, налоги)

Инвестиционные расходы (оборудование, ПО)

Сальдо на конец месяца

Рассчитайте cash runway (на сколько месяцев хватит денег) при текущем остатке, например, 2 000 000 руб.

Назовите 3 финансовых метрики для стартапа (не CAC/LTV, а другие, например, gross margin, burn rate, месячный рост выручки)

Составьте 3 сценария финансовой модели: пессимистичный, реалистичный, оптимистичный (отличаются по выручке и расходам) — таблица на 6 месяцев

Напишите чек-лист финансового контроля на каждый месяц (5 пунктов, что проверять и контролировать)

Эталон выполнения

Cash Flow на 6 месяцев (таблица)

Cash runway (расчёт)

3 финансовые метрики (с пояснением)

3 сценария (пессимистичный, реалистичный, оптимистичный)

Чек-лист финансового контроля (5 пунктов)

Критерии оценки

Что проверяется	Балл
Cash Flow на 6 месяцев	1
Cash runway рассчитан	1
3 финансовые метрики названы	1
3 сценария построены	1
Чек-лист финансового контроля	1
Итого	5

Задание 4. Масштабирование и защита бизнеса

Условие

Разработайте стратегию масштабирования и защиты вашего бизнеса.

Требования:

Опишите модель масштабирования: органический рост, франчайзинг, лицензирование, выход в новые регионы, партнёрства (выберите 1–2 и обоснуйте)

Назовите 3 юридических аспекта масштабирования (регистрация в новом регионе, налоги, защита ИС, договоры с партнёрами, трудовое законодательство)

Составьте карту рисков масштабирования (минимум 5 рисков) и для каждого — план смягчения

Опишите способы защиты бизнеса:

Интеллектуальная собственность (товарный знак, патент, авторское право)

Договоры с контрагентами (NDA, оферта, договор услуг)

Безопасность данных (152-ФЗ, хранение персональных данных)

Напишите план защиты бизнеса на первый год (пошагово, что сделать в первую очередь)

Эталон выполнения

Модель масштабирования с обоснованием

3 юридических аспекта

Карта рисков (5+ рисков + смягчение)

Способы защиты бизнеса

План защиты на первый год

Критерии оценки

Что проверяется	Балл
Модель масштабирования обоснована	1
3 юридических аспекта названы	1
Карта рисков (5+ рисков)	1
Способы защиты бизнеса описаны	1
План защиты на первый год	1
Итого	5

Пример оценочных материалов для итоговой аттестации

Часть 1. Теоретическая часть (письменные ответы)

Формат: слушатель письменно отвечает на 20 вопросов. Каждый ответ должен быть содержательным (3–5 предложений, чётко по существу). Максимум — 20 баллов (1 балл за каждый полный и правильный ответ).

Модуль 1. Основы IT-инфраструктуры

Вопрос 1. Какие основные команды Linux используются для навигации по файловой системе, просмотра содержимого каталогов и управления файлами? Опишите их назначение.

Вопрос 2. Что такое Git? Опишите базовый цикл работы с Git (init, add, commit, push) и объясните, зачем нужна каждая команда.

Вопрос 3. Что такое Docker? Для чего используются Dockerfile и команды docker build и docker run?

Модуль 2. Алгоритмизация и структурное программирование

Вопрос 4. Что такое указатель в языке C? Как происходит работа с динамической памятью (malloc, free)? Почему важно освобождать память?

Вопрос 5. Объясните, как работает пузырьковая сортировка (bubble sort). Какова её временная сложность в лучшем и худшем случаях?

Вопрос 6. Что такое рекурсия? Приведите пример рекурсивной функции и объясните, что такое «базовый случай» и почему он необходим.

Модуль 3. Объектно-ориентированное программирование

Вопрос 7. Объясните три основных принципа ООП: инкапсуляция, наследование, полиморфизм. Приведите примеры.

Вопрос 8. Что такое перегрузка операторов в C++? Для чего она используется? Приведите пример перегрузки оператора + для класса Matrix.

Вопрос 9. Что такое абстрактный класс и чисто виртуальные методы? Когда и зачем их используют?

Модуль 4. Современная прикладная разработка

Вопрос 10. Что такое JWT (JSON Web Token) и как он используется для авторизации в веб-приложениях? Опишите структуру токена.

Вопрос 11. Назовите и кратко охарактеризуйте основные парадигмы программирования: процедурную, объектно-ориентированную, функциональную. В чём их различия?

Модуль 5. Инфраструктурные решения и управление данными

Вопрос 12. Какие типы JOIN существуют в SQL? Объясните разницу между INNER JOIN, LEFT JOIN, RIGHT JOIN. Приведите примеры.

Вопрос 13. Что такое агрегационные функции в SQL (COUNT, SUM, AVG, MIN, MAX)? Для чего используется GROUP BY и HAVING?

Модуль 6. Старт бизнеса

Вопрос 14. Что такое Lean Startup? Объясните цикл Build → Measure → Learn и понятие MVP (Minimum Viable Product).

Вопрос 15. Что такое Custdev (Customer Development) и как с его помощью проверять гипотезы? Опишите основные принципы проведения интервью с потенциальными клиентами.

Модуль 7. Рост бизнеса

Вопрос 16. Что такое юнит-экономика? Объясните метрики CAC, LTV, ARPU. Какое соотношение LTV / CAC считается здоровым для бизнеса?

Вопрос 17. Что такое продуктовый подход и работа с гипотезами? Как приоритизировать гипотезы (например, методом ICE или RICE)?

Модуль 8. Продуктовый подход в бизнесе

Вопрос 18. Что такое стратегия голубого океана? Опишите инструменты: стратегическая канва и матрица «убрать — уменьшить — повысить — создать».

Вопрос 19. Что такое подрывные инновации (disruptive innovation) по Кристенсену? Приведите пример и объясните, как стартап может победить лидера рынка.

Модуль 9. Азбука стартапа

Вопрос 20. Какие основные налоговые режимы существуют для малого бизнеса (например, УСН 6%, УСН 15%, НПД)? В чём их различия и как выбрать подходящий для стартапа?

Часть 2. Практическая часть

Форма выполнения: письменная. Слушатель выполняет одно задание по выбору (5 вариантов). Код пишет самостоятельно. В эталоне — только описание требований и критерии.

Задания

Вариант А. Структурное программирование на C

Разработайте программу на языке C, которая:

Принимает из командной строки имя входного файла и имя выходного файла

Читает матрицу вещественных чисел из входного файла (формат: первая строка — rows cols, затем rows строк по cols чисел)

Вычисляет сумму всех элементов матрицы, находит минимальный и максимальный элемент

Записывает результат в выходной файл

Освобождает всю динамически выделенную память

Вариант Б. Объектно-ориентированное программирование на C++

Создайте шаблонный класс DynamicArray с методами: push_back, pop_back, size, at (с проверкой границ и выбросом исключения), оператор []. Продемонстрируйте работу всех методов в функции main.

Вариант В. Базы данных и SQL

Напишите 5 SQL-запросов для базы данных интернет-магазина (таблицы products и orders):

Товары с ценой от 1000 до 5000

Количество проданных единиц по каждому товару

Топ-3 самых продаваемых товара

Товары, которые ни разу не заказывались

Общая выручка за все время

Вариант Г. Прикладная разработка (язык по выбору)

Разработайте консольный Todo-менеджер на выбранном языке (Python, C#, Java, JavaScript, Go, Kotlin, Swift) с операциями:

Добавить задачу (название, описание, статус: новая / в работе / выполнена)

Показать все задачи

Показать задачи по статусу

Отметить задачу как выполненную

Удалить задачу

Сохранить задачи в файл

Загрузить задачи из файла

Используйте ООП (класс Task).

Вариант Д. DevOps и Linux

Опишите пошагово (с командами и пояснениями) выполнение следующих задач:

Установка Linux на виртуальную машину

Создание пользователя admin и добавление его в группу sudo

Настройка статического IP-адреса
Установка Docker
Запуск контейнера с Nginx (проброс порта 8080)
Написание Dockerfile для приложения, выводящего "Hello, DevOps!"

Вариант Е. Бизнес-план стартапа (предпринимательство)
Разработайте бизнес-план для технологического стартапа (идею выберите самостоятельно).
Требования:
Опишите бизнес-идею (продукт, целевая аудитория, проблема)
Сформулируйте гипотезу ценности и гипотезу роста
Опишите MVP (минимально жизнеспособный продукт)
Проведите анализ рынка (TAM, SAM, SOM с обоснованием)
Рассчитайте юнит-экономику (CAC, LTV, ARPU, соотношение LTV/CAC)
Опишите стратегию привлечения первых клиентов
Назовите 3 основных риска и способы их снижения
Формат сдачи: аналитический отчёт (до 5 страниц) или презентация (до 10 слайдов).

Вариант Ж. Стратегия стартапа (предпринимательство)
Выберите реальный стартап (или создайте свой) и разработайте стратегию развития.
Требования:
Краткое описание стартапа (продукт, рынок, конкуренты)
Примените стратегию голубого океана: заполните матрицу «убрать — уменьшить — повысить — создать»
Постройте стратегическую канву (кривую ценности) для вашего продукта и конкурентов
Сформулируйте стратегию подрыва (low-end или new-market disruption) — как победить лидера
Опишите факторы успеха стартапа (команда, продукт, рынок)
Составьте roadmap на первый год (по кварталам)
Формат сдачи: аналитический отчёт или презентация.

Вариант З. Франшиза и масштабирование (предпринимательство)
Проведите анализ открытия бизнеса по франшизе.
Требования:
Выберите 3 реальные франшизы из одной ниши (например, общепит, образование, услуги)
Составьте сравнительную таблицу (паушальный взнос, роялти, инвестиции, срок окупаемости)
Выберите лучшую франшизу и обоснуйте выбор
Рассчитайте примерную прибыль и точку безубыточности
Опишите план масштабирования: как открыть вторую и третью точку
Назовите 3 юридических аспекта, которые нужно проверить перед покупкой франшизы
Составьте чек-лист вопросов к франчайзору (минимум 10 вопросов)
Формат сдачи: аналитический отчёт с таблицами и расчётами.
Критерии оценивая и шкалы приведены в п.2.5.1

Промежуточная аттестация проводится в форме экзамена по модулю в виде выполнения практического здания по пройденным темам. Для успешного прохождения ПА слушатель выполняет по одной индивидуальной работе по каждому из модулей и

размещает полный комплект материалов в ЛМС, согласно утверждённым критериям. Формат демонстрации не предусмотрен.

Критерии оценивания промежуточной аттестации:

– соответствие заданию модуля и полнота выполнения всех пунктов;
– корректность решений и валидность результатов;
– качество оформления;
– соблюдение академической добросовестности (оригинальность, корректные заимствования, отсутствие совпадений, не объяснённых ссылками);
– соблюдение сроков сдачи в ЛМС.

Шкалы оценивания:

Баллы/Критерии	**соответствие заданию модуля и полнота выполнения всех пунктов**	**корректность решений и валидность результатов;**	**качество оформления;**	**соблюдение академической добросовестности**	**соблюдение сроков сдачи в ЛМС.**
5	Выполнены все пункты задания модуля в полном объёме, нет пропусков, работа полностью соответствует заданию	Решения технически/логически верны, результаты воспроизводимы и обоснованы, обработка ошибок корректна	Структурировано, читаемо, единый стиль, код/текст отформатированы, есть комментарии/пояснения, файлы названы логично, материалы упакованы аккуратно	Работа полностью оригинальна (или все заимствования явно указаны ссылками), нет совпадений с другими работами, нет плагиата	--
4	Выполнены все основные пункты, но отсутствует 1 незначительный элемент / незначительная часть	Решения в целом верны, но есть 1–2 незначительные ошибки / неточности, не влияющие на основной результат	Хорошее оформление, но есть мелкие недочёты: местами нарушен стиль, не все файлы подписаны, отсутствуют отдельные комментарии	Единичные совпадения, объяснённые корректными ссылками; общая оригинальность высокая (>80%)	--
3	Выполнено 70–80% задания, отсутствуют 2–3 пункта или один значительный блок	Решения работают, но есть ошибки в краевых случаях / не обработаны исключения / результаты частично неверны	Среднее качество: структура есть, но оформление «грязное» (лишние файлы, разный стиль, мало комментариев)	Есть заимствования без ссылок (10–20% работы), или незначительные совпадения с другими слушателями	Работа сдана строго в установленный срок (включая все материалы)
2	Выполнено 50–69% задания, отсутствует более 3 пунктов или ключевая часть	Решения содержат системные ошибки, результаты невалидны или не воспроизводятся	Оформление плохое: трудно читать, нет структуры, неясно, что к чему относится	Значительные заимствования без ссылок (20–40%), или заметные совпадения с другими работами	Незначительная задержка (до 3 календарных дней)
1	Выполнено менее 50% задания, работа не соответствует	Решения не работают, результаты отсутствуют или	Оформление отсутствует (набор файлов без пояснений,	Более 40% работы скопировано (без ссылок), или	Задержка от 4 до 7 календарных дней

	заданию	противоречивы	нечитаемый код/текст)	работа несамостоятельна	
0	Работа не сдана или не относится к модулю	Решения не представлены или полностью неверны	Материалы не загружены в ЛМС или загружены с нарушением формата (не открываются)	Полный плагиат (работа целиком не своя, или копия другой работы)	Задержка более 7 календарных дней или работа не сдана

Итоговая шкала оценивания

Отлично	21–23 балла
Хорошо	17–20 баллов
Удовлетворительно	14–16 баллов
Неудовлетворительно	менее 14 баллов

Слушателям, получившие по результатам промежуточной аттестации оценку «неудовлетворительно» устанавливаются сроки повторной промежуточной аттестации. Если обучающийся не ликвидировал академическую задолженность при прохождении повторной промежуточной аттестации в первый раз, ему предоставляется возможность пройти повторную промежуточную аттестацию во второй раз.

Первая повторная промежуточная аттестация и вторая повторная промежуточная аттестация проводятся комиссией. Состав комиссии формируется департаментом образовательных программ.

Итоговая аттестация (ИА): в рамках итоговой аттестации проводятся итоговые экзамены на знание теоретической части и выполнение практических заданий.

Критерии оценивания:

Теоретическая часть:

– Правильность ответов — процент верных ответов от общего числа вопросов
– Полнота охвата — отвечены ли вопросы по всем ключевым темам модуля

Практическая часть:

– Соответствие заданию — выполнение всех пунктов задания
– Корректность решения — работоспособность, отсутствие ошибок
– Валидность результатов — результаты верны и воспроизводимы
– Качество кода/решения — читаемость, структура, комментарии, обработка ошибок
– Оформление — инструкции, логичная структура материалов

Общие требования (для теории и практики):

– Соблюдение формата — требования к объёму, времени, способу сдачи
– Академическая добросовестность — отсутствие плагиата, списывания, подсказок

Шкалы оценивания:

Баллы/Критерии	5	4	3	2	1	0
Правильность ответов	90–100% верных ответов	75–89% верных ответов	50–74% верных ответов	менее 50% верных отвтеов	—	—
Полнота охвата	Ответы охватывают все ключевые темы курса	Ответы охватывают основные темы, но есть 1–2 пропущенных раздела	Ответы охватывают менее 70% ключевых тем	Ответы не охватывают ключевые темы или тест отсутствует	—	—

Соответствие заданию	Выполнены все пункты задания	Выполнены все основные, пропущен 1 незначительный	Выполнено 70–80%	Выполнено менее 70%	—	—
Корректность решения	Решение верно, ошибок нет	1–2 незначительные ошибки	Решение работает, но есть ошибки	Решение не работает или содержит системные ошибки	—	—
Валидность результатов	Результаты верны, воспроизводимы, обоснованы	В целом верны, но 1–2 результата не обоснованы	Результаты частично верны или частично не воспроизводятся	Результаты отсутствуют или неверны	—	—
Качество реализации	Код/решение структурированы, читаемы, есть комментарии, обработка ошибок	Хорошо, но есть мелкие недочёты	Удовлетворительно, но есть «грязь»	Нечитаемо/не компилируемо	—	—
Оформление	Есть инструкция, логичные названия, аккуратная упаковка	Хорошо, но есть 1–2 недочёта	Средне: что-то есть, но неполно	Оформление отсутствует	—	—
Соблюдение формата	—	—		Существенные отклонения Минус 2 балла	Незначительные отклонения Минус 1 балл	Формат полностью соблюдён 0 баллов
Академическая добросовестность	—		Более 30% заимствований или копирование чужой работы – работа не принимается	Значительные заимствования (10–30%) Минус 2 балла	Незначительные заимствования (до 10%) Минус 1 балл	Ответ/решение оригинальны 0 баллов

Итоговая шкала оценивания

Отлично	45 – 50 баллов
Хорошо	35 – 44 балла
Удовлетворительно	25 – 34 балла
Неудовлетворительно	0 – 24 балла

Лица, не прошедшие итоговую аттестацию или получившие на итоговой аттестации неудовлетворительные результаты, вправе повторно пройти итоговую аттестацию в сроки, определяемые академией.

2.4. Методические материалы

Методические рекомендации для работы на теоретических занятиях

Теоретическое занятие является одной из основных системообразующих форм организации образовательного процесса, представляющее собой систематическое, последовательное, монологическое изложение педагогическим работником - лектором учебного материала теоретического характера. Лекция представляет собой элемент технологии представления учебного материала путем логически стройного, систематически последовательного и ясного изложения, целью которого является организация целенаправленной познавательной деятельности обучающихся (слушателей) по овладению программным материалом. Изучение дисциплины требует систематического и последовательного накопления знаний, поэтому слушателям рекомендуется перед очередной лекцией просмотреть по конспекту материал предыдущей. При затруднениях в восприятии материала следует обращаться к основным литературным источникам, лектору или интернет источнику, предложенному в списке.

Методические рекомендации для работы на практических занятиях

Важной составной частью образовательного процесса являются практические занятия. Практические занятия – организационная форма, в процессе которой обучающиеся самостоятельно изучают учебный материал по различным источникам знаний и коллективно обсуждают результаты своей работы, выполняя задания, предложенные преподавателем. Практические занятия проводятся по темам, требующим научно - теоретического обобщения литературных источников, и помогают обучающимся глубже усвоить учебный материал, приобрести навыки работы с нормативными документами и научно-методической литературой.

Эффективность занятий практического типа во многом зависит от качества предшествующих лекционных занятий и самоподготовки обучающихся. Темы практических занятий и рекомендуемая литература, цель и задачи ее изучения сообщаются педагогическим работником на вводных занятиях каждого раздела. Начиная подготовку к практическому занятию, преподавателю необходимо рекомендовать обучающемуся поработать с дополнительной литературой, сделать записи по рекомендованным источникам. Необходимо помнить, что на лекционном занятии материал рассматривается не в полном объёме. Остальная его часть восполняется в процессе практического занятия. Особое внимание при этом необходимо обратить на суть основных положений и выводов, объяснение явлений и фактов, уяснение практического приложения рассматриваемых теоретических вопросов. При необходимости следует обращаться за консультацией к педагогическому работнику. На практическом занятии каждый обучающийся должен быть готовым к выступлению по всем поставленным преподавателем вопросам, проявлять максимальную активность при рассмотрении кейсов, проблемных ситуаций и др.

Методические рекомендации обучающимся (слушателям) по организации самостоятельной работы

Самостоятельная работа обучающихся (слушателей) – деятельность, которая выполняется без непосредственного участия педагогического работника, но под его руководством. Обучающийся (слушатель), обладающий навыками самостоятельной работы, активнее и глубже усваивает учебный материал, оказывается лучше подготовленным к творческому труду, к самообразованию и продолжению обучения. Обучающийся знакомится с дидактическим материалом, отвечает на вопросы самоконтроля, выполняет задания творческого характера. Своевременное и качественное выполнение самостоятельной работы

базируется на соблюдении настоящих рекомендаций и изучении рекомендованной литературы. Обучающийся может дополнить список использованной литературы современными источниками, не представленными в списке рекомендованной литературы, и в дальнейшем использовать собственные подготовленные учебные материалы при подготовке к зачету, а также выступлениям на методических и педагогических советах ОО.

Групповая и индивидуальная консультация

Разъяснение является основной формой занятий по рассмотрению наиболее сложных вопросов изучаемого программного материала. Цель консультации – максимальное приближение обучения к практическим интересам с учетом имеющейся информации и является результативным материалом закрепления знаний.

Индивидуальная консультация – это совместная работа обучающихся

(слушателей) с педагогическим работником. Цель индивидуальной консультации – помощь обучающимся в решении сложных вопросов, возникающих при освоении ДПП ПК. На индивидуальной консультации обучающийся (слушатель) совместно с преподавателем подробно разбирает проблему или ситуацию, с которой он обратился за помощью. Преподаватель помогает глубинно проработать проблемный вопрос. Групповая консультация проводится в следующих случаях:

• когда необходимо подробно рассмотреть практические вопросы, которые были недостаточно освещены или совсем не освещены в процессе лекции;

• с целью оказания помощи в самостоятельной работе (написание научно-методической статьи, подготовки выступления к научно-практической конференций и другим мероприятиям по заявленной проблематике);

• если обучающиеся самостоятельно изучают нормативный, справочный материал, инструкции, положения.

3. Организационно-педагогические условия реализации программы

3.1. Материально-технические условия реализации программы

Условия для функционирования электронной информационно-образовательной среды

Электронные информационные и образовательные ресурсы	Вид занятий	Наименование оборудования, программного обеспечения, необходимого слушателю для пользования ЭИОС
Система дистанционного обучения GetCourse, система видеоконференцсвязи	Лекции, Лабораторные работы, Практические занятия, СРС	Компьютер, подключенный к сети Интернет; интернет-браузер; наушники, микрофон, веб-камера, специализированное программное обеспечение, цифровые сервисы

3.2. Учебно-методическое и информационное обеспечение программы

Учебно-методические материалы программы размещены в электронном курсе в СДО GetCourse. В электронном курсе реализована система обратной связи в формате форума.

Перечень средств обучения

Перечень необходимого ПО

ПО	Mac	Linux

Visual Studio Code	Newest	Newest
GCC	13.2.1	13.2.1
clang-format	17.0.5	17.0.5
libncurses5-dev	6.5	6.5
pcre	8.45	8.45
regex	-	-
valgrind	-	3.22.0
cppcheck	2.12.1	2.12.1
check	0.15.2	0.15.2
iTerm2	Newest	-
Docker	Newest	Newest
Git	Newest	Newest
CEF	0118.6.9	0118.6.9
Qt	6.7	6.7
GTK+	4.12.1	4.12.1
raygui	3.6	3.6
Gtest	1.14.0	1.14.0
dockle	Newest	Newest
VirtualBox	Newest	Newest
Wireshark	Newest	Newest
Apache Tomcat	9.0.89	9.0.89
JMeter	5.6.3	5.6.3
Elasticsearch	8.13	8.13
Postman	Newest	Newest
iperf	Newest	Newest
nginx	1.26.0	1.26.0
spawn-fcgi	1.6.4	1.6.4
fcgi	2.4.2	2.4.2
Python	3.12.3	3.12.3
PyCharm	Newest	Newest
.NET В том числе Asp. Net Core	8.0.4	8.0.4
Dotnet SDK	8.0.204	8.0.204
dotnet-ef	-	-
Oracle JDK / OpenJDK	22	22
Maven 3	3.9.6	3.9.6
VisualVM	Newest	Newest
JUnit	4.13.2	4.13.2
Java	17	17
IntelliJ IDEA	Newest	Newest
mockito-kotlin	5.3.1	5.3.1
kotlin.test	1.9	1.9
kotlinx.coroutines	1.8.0	1.8.0
Moxy	2.2.2	2.2.2
Ktor	2.3.10	2.3.10

Kotlin Exposed	0.5	0.5
SQLite	3.45.3	3.45.3
Android Studio	Newest	Newest
Android SDK	34	34
H2 Database Android	2.2.224	2.2.224
Retrofit 2	2.11.0	2.11.0
Dagger 2	2.51.1	2.51.1
Gradle	8.2	8.2
iOS SDK	Newest	-
CocoPods	Newest	Newest
SwiftUI	Newest	-
Node	20.13.0	20.13.0
Yarn	4.2.1	4.2.1
React Devtools	Newest	Newest
Angular Dev Tools	Newest	Newest
Vue Dev tools	Newest	Newest
Redux Dev tools	Newest	Newest
Go	1.22.3	1.22.3
DBeaver	Newest	Newest
PostgreSQL	14.2	14.2
pcre2	10.42	10.42

3.3. Литература

3.2.1. Основная литература

Git для профессионального программиста. Чакон Скотт, Штрауб Бен (https://git-scm.com/book/ru/v2)

Linux. Командная строка. Лучшие практики. Дэниел Джей Барретт

Linux. Карманный справочник. Граннеман Скотт

Изучаем vi и Vim. Не просто редакторы. Ханна Элберт, Роббинс Арнольд

Язык программирования Си. Брайан Керниган, Деннис Ритчи.

Полный справочник по Си. Герберт Шилдт

Программирование на С для начинающих. Грег Перри, Дин Миллер

Язык С. Справочник. Полное описание языка. Принц Питер, Кроуфорд Тони

3.2.2. Дополнительная литература

Грокаем алгоритмы. Адитья Бхаргава

Алгоритмы для начинающих. Луридас Панос

Computer Science для программиста-самоучки. Все что нужно знать о структурах данных и алгоритмах. Альтхофф Кори

Чистый код: создание, анализ и рефакторинг. Мартин Роберт

Идеальный программист. Как стать профессионалом разработки ПО. Мартин Роберт

3.2.3. Интернет-ресурсы

1. Официальная документация Python (русская версия): https://docs.python.org/ru/3/
2. Scikit-learn — документация и примеры: https://scikit-learn.org/stable/
3. Kaggle Learn (курсы по Python, pandas, ML): https://www.kaggle.com/learn
4. Google Colab: https://colab.research.google.com/

5. Real Python: https://realpython.com/

6. Хабр / Хабр Карьера — разделы «Python», «Машинное обучение»: https://habr.com/ru/hub/python/; https://habr.com/ru/hub/machine_learning/

7. Open Data Science (ods.ai): https://ods.ai/

8. eLIBRARY.RU: https://elibrary.ru/

3.4. Кадровое обеспечение программы

Реализация образовательной программы обеспечивается квалифицированными педагогическими кадрами, соответствующими требованиям профессионального стандарта «Педагог профессионального обучения, профессионального образования и дополнительного профессионального образования».

Требования к квалификации

Категория преподавателей	**Требования**
Преподаватели технических модулей (C, C++, C#, Java, Python, Go, Swift, Kotlin, алгоритмы, структуры данных, DevOps, базы данных)	Высшее образование по IT-направлению (прикладная информатика, программная инженерия, компьютерные науки) + опыт разработки или преподавания не менее 2 лет
Преподаватели предпринимательских модулей (Lean Startup, Custdev, анализ рынка, юнит-экономика, стратегии)	Высшее образование (экономическое / управленческое / маркетинговое) + опыт в предпринимательстве или бизнес-консалтинге не менее 2 лет
Преподаватели машинного обучения (при наличии модуля)	Высшее образование (Data Science, математика, ИТ) + опыт работы с библиотеками ML (scikit-learn, TensorFlow, Keras) не менее 1 года
Преподаватели предпринимательских модулей	Высшее образование (экономическое / управленческое / маркетинговое / предпринимательское) + опыт в предпринимательстве, запуске стартапов, бизнес-консалтинге или управлении проектами не менее 2 лет

Доля преподавателей с практическим опытом

Не менее 70% преподавателей, реализующих программу, имеют практический опыт работы в IT-индустрии или предпринимательстве не менее 2 лет.

Не менее 30% преподавателей имеют опыт участия в реальных коммерческих проектах.

Повышение квалификации

Преподаватели проходят повышение квалификации не реже одного раза в 3 года

Формы повышения квалификации: курсы по технологиям (обновление стека), стажировки в IT-компаниях, участие в профильных конференциях, обучение педагогическим компетенциям.