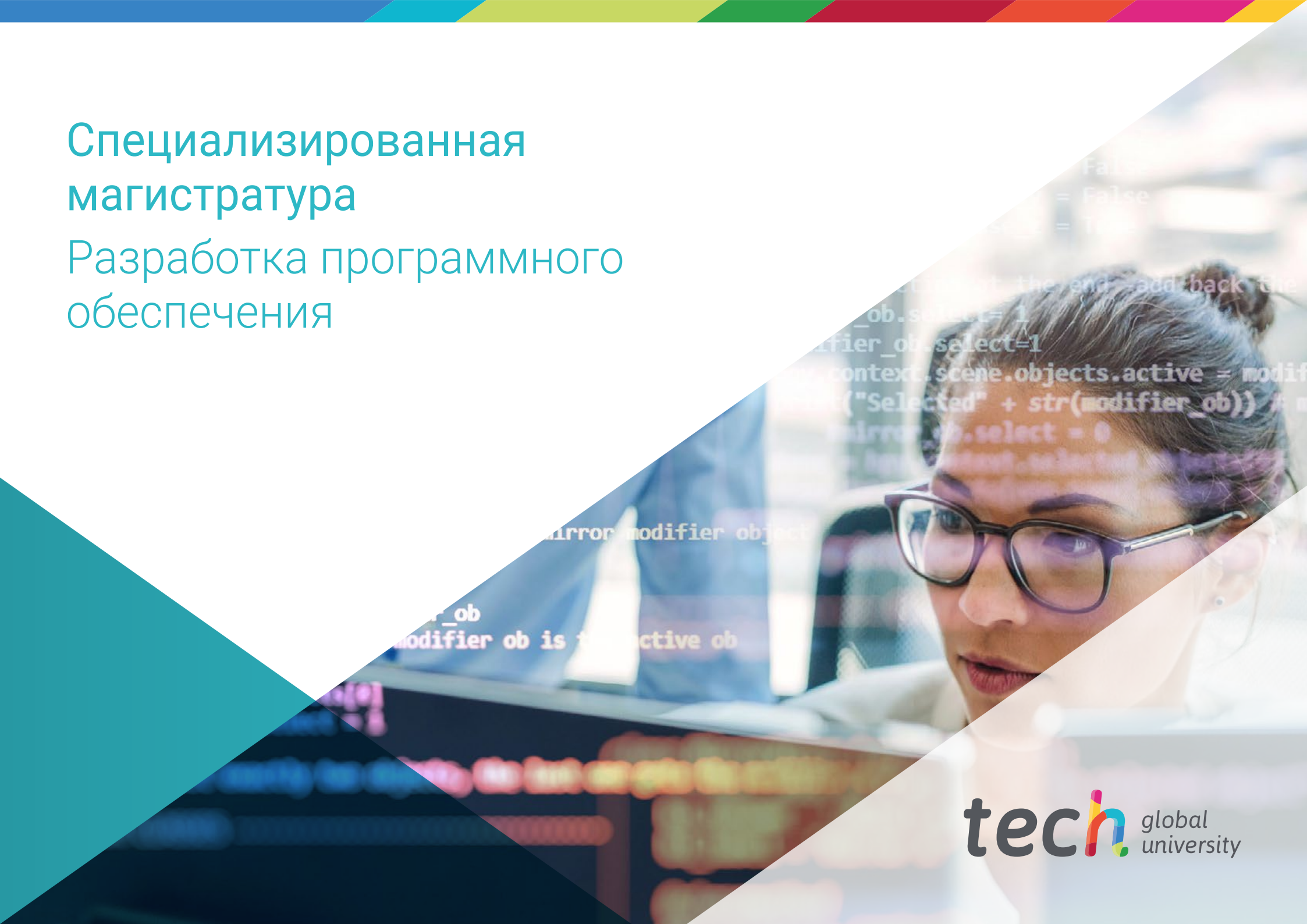
Специализированная магистратура

Разработка программного обеспечения



tech global university

tech global university

Специализированная магистратура
Разработка программного обеспечения

- Формат: **онлайн**
- Продолжительность: **12 месяцев**
- Учебное заведение: **TECH Global University**
- Аккредитация: **60 ECTS**
- Расписание: **по своему усмотрению**
- Экзамены: **онлайн**

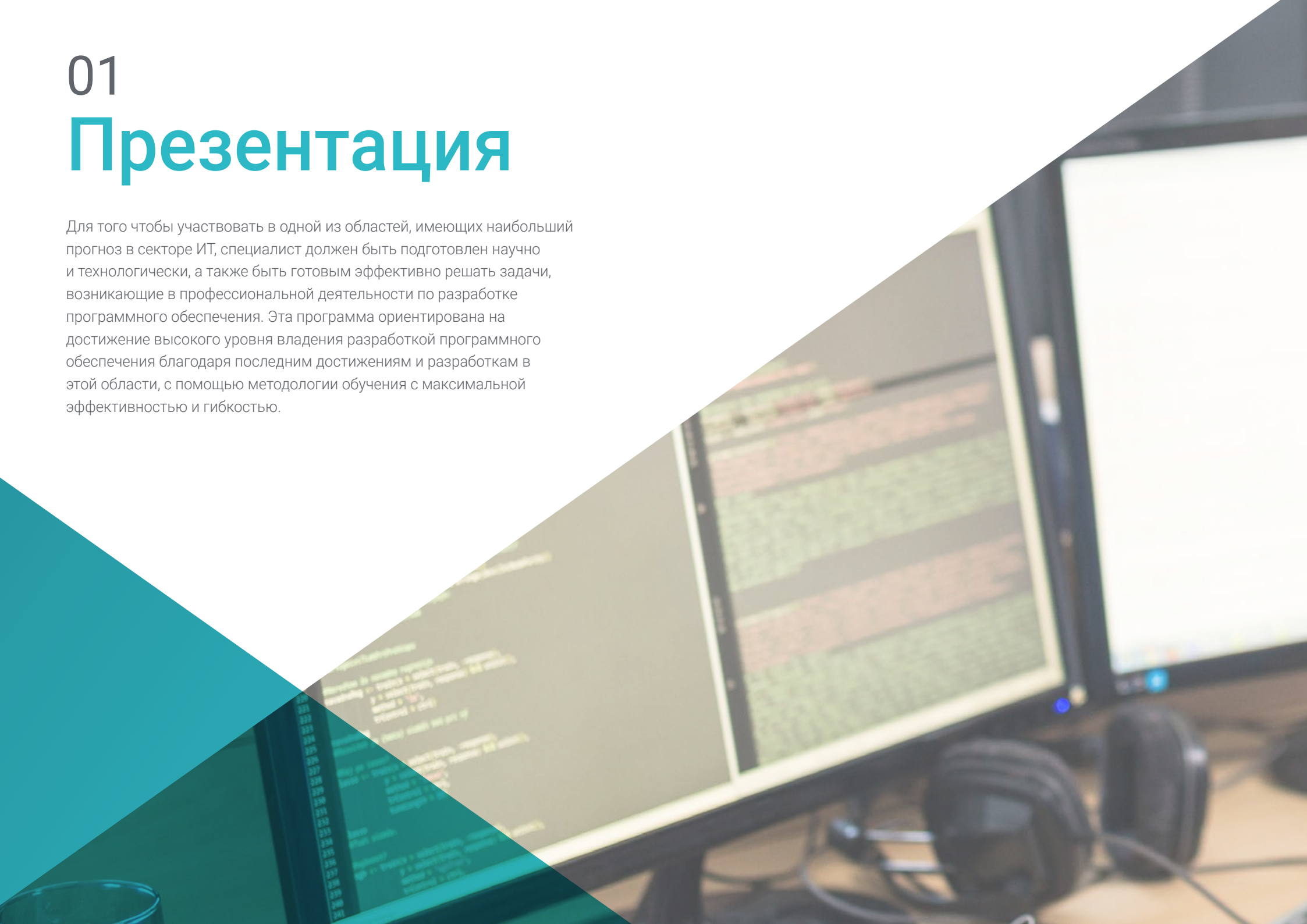
Веб-доступ: www.techtitute.com/ru/information-technology/professional-master-degree/master-software-development

Оглавление

01

Презентация

Для того чтобы участвовать в одной из областей, имеющих наибольший прогноз в секторе ИТ, специалист должен быть подготовлен научно и технологически, а также быть готовым эффективно решать задачи, возникающие в профессиональной деятельности по разработке программного обеспечения. Эта программа ориентирована на достижение высокого уровня владения разработкой программного обеспечения благодаря последним достижениям и разработкам в этой области, с помощью методологии обучения с максимальной эффективностью и гибкостью.



Приобретите наиболее полные знания в области программной инженерии на самом современном обучении на рынке онлайн-образования и начните работать над разработками в этой динамичной профессиональной области"

С развитием новых технологий программное обеспечение стало крайне важным элементом в современном мире. В последние годы стала очевидной необходимость уметь разрабатывать программные продукты с нужной функциональностью и качеством, в срок и в рамках бюджета.

Эта программа предназначена для тех, кто заинтересован в достижении более высокого уровня знаний в области разработки программного обеспечения. Основной целью является обучение студентов, чтобы они могли применить знания, полученные в этой Специализированной магистратуре, в реальном мире, в рабочей среде, воспроизводящей условия, с которыми они могут столкнуться в будущем, в строгой и реалистичной манере.

Воспользуйтесь возможностью пройти обучение в 100% онлайн-формате, не отказываясь от своих обязанностей и легко проходя обучение в университете. Обновите свои знания и получите степень Специализированной магистратуры, чтобы продолжать расти личностно и профессионально.

Вы получите обширные знания в области программной инженерии, а также в области информатики и компьютерной структуры, включая математические, статистические и физические основы, необходимые в инженерии.

Воспользуйтесь возможностью пройти обучение в 100% онлайн-формате, не отказываясь от своих обязанностей и легко проходя обучение в университете. Обновите свои знания и получите степень Специализированной магистратуры, чтобы продолжать расти личностно и профессионально.

Данная **Специализированная магистратура в области разработки программного обеспечения** содержит самую полную и современную образовательную программу на рынке. Основными особенностями обучения являются:

- Разработка более 100 смоделированных сценариев, представленных экспертами в разработке программного обеспечения
- Графическое, схематичное и исключительно практичное содержание программы предоставляет научную и практическую информацию о разработке программного обеспечения
- Новости о последних достижениях в области разработки программного обеспечения
- Практические упражнения, в которых процесс самоконтроля может быть использован для улучшения эффективности обучения
- Интерактивная система обучения на основе кейс-метода и его применение в реальной практике
- Все вышеперечисленное дополнят теоретические занятия, вопросы к эксперту, дискуссионные форумы по спорным вопросам и индивидуальная работа по закреплению материала
- Доступ к учебным материалам с любого стационарного или мобильного устройства с выходом в интернет

Эта программа позволит вам узнать о базовой структуре компьютера и его программном обеспечении, что послужит основой для повышения вашей квалификации"

Узнайте все необходимое для безопасной работы с языками программирования, включив в свои знания интерпретацию и разработку основных алгоритмов для работы в программировании"

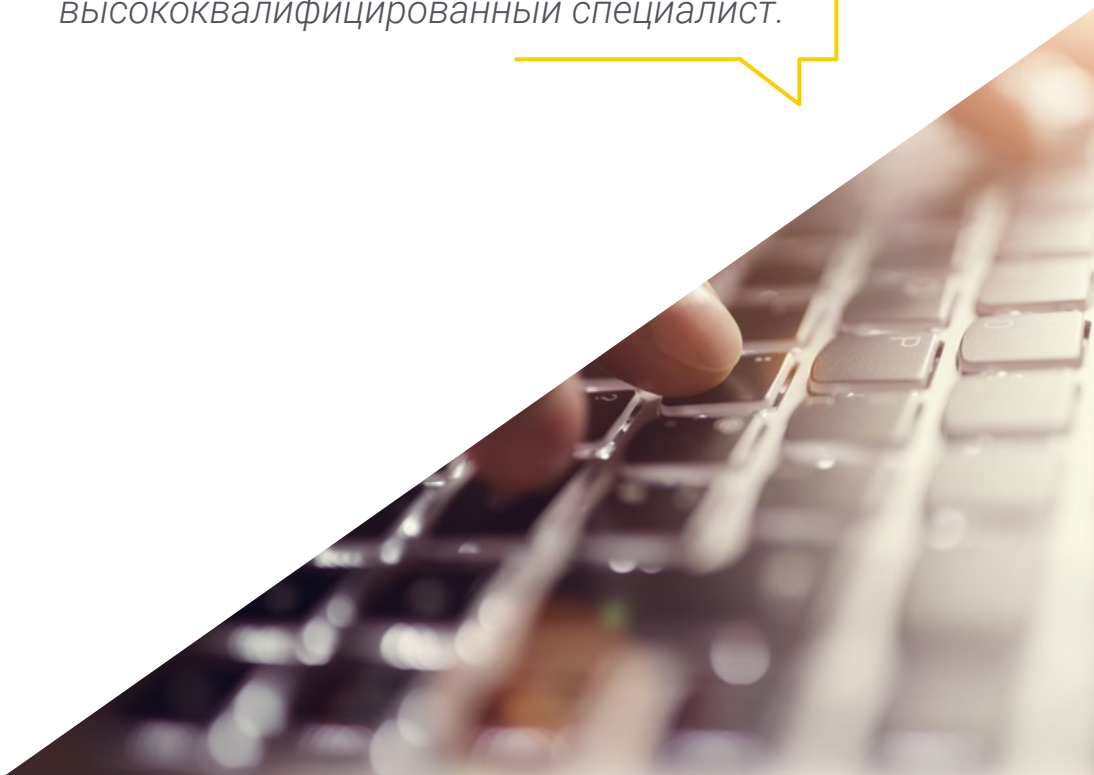
Обучение, которое позволит вам понять, как работает компьютерная программа и как вмешаться во все основные элементы компьютерной программы.

Познакомьтесь с новейшими системами обработки данных, представленными на рынке, научитесь разрабатывать передовые алгоритмы и изучите все аспекты, которыми должен владеть высококвалифицированный специалист.

В преподавательский состав входят профессионалы из мира разработки программного обеспечения, которые привносят в обучение свой опыт работы, а также признанные специалисты из ведущих сообществ и престижных университетов.

Мультимедийное содержание программы, разработанное с использованием новейших образовательных технологий, позволит профессионалам проходить обучение в симулированной среде, обеспечивающей иммерсивный учебный процесс, основанный на обучении в реальных ситуациях.

Структура этой программы основана на проблемно-ориентированном обучении, с помощью которого преподаватель должен попытаться решить различные ситуации из профессиональной практики, возникающие в ходе программы. В этом специалисту будет помогать инновационная интерактивная видеосистема, созданная признанными экспертами в области разработки программного обеспечения с большим опытом преподавания.



02

Цели

Цель данного обучения - предоставить специалистам, работающим в области разработки программного обеспечения, знания и навыки, необходимые для осуществления своей деятельности, с использованием самых современных протоколов и методик. Благодаря подходу к работе, который полностью адаптируется под студента, эта программа Специализированной магистратуры постепенно приведет вас к приобретению навыков, которые продвинут вас на более высокий профессиональный уровень.

Вы погрузитесь в область вычислений и компьютерной структуры - важнейших предметов для любого разработчика программного обеспечения"

Общие цели

- Обучить студента в научном и технологическом плане, а также подготовить к профессиональной практике в области программной инженерии, все это с помощью комплексной и разносторонней программы, адаптированной к новым технологиям и инновациям в этой области
- Получить обширные знания в области программной инженерии, а также в области информатики и компьютерной структуры, включая математические, статистические и физические основы, необходимые в инженерии

Достигните желаемого уровня знаний и овладейте разработкой программного обеспечения с помощью этого обучения высшего уровня"

Конкретные цели

Модуль 1. Основы программирования

- Понимать базовую структуру компьютера, программное обеспечение и языки программирования общего назначения
- Научиться разрабатывать и интерпретировать алгоритмы, которые являются необходимой основой для разработки программного обеспечения
- Понимать основные элементы компьютерной программы, такие как различные типы данных, операторы, выражения, утверждения, операторы ввода-вывода и управления
- Понимать различные структуры данных, доступные в языках программирования общего назначения, как статических, так и динамических, и приобрести необходимые знания по работе с файлами
- Знать различные методы тестирования программного обеспечения и важность создания хорошей документации вместе с хорошим исходным кодом
- Изучить основы языка программирования C++, одного из самых распространенных языков программирования в мире

Модуль 2. Структура данных

- Изучить основы программирования на C++, включая классы, переменные, условные выражения и объекты
- Понимать абстрактные типы данных, линейные типы структур данных, простые и сложные иерархические структуры данных и их реализацию на C++
- Понимать работы расширенных структур данных, отличающихся от обычных
- Изучить теорию и практику, связанные с использованием структур данных кучи и приоритетных очередей
- Узнать, как работают таблицы *Hash* в качестве абстрактных типов данных и функций
- Понять теорию графов, а также продвинутые алгоритмы и концепции графов

Модуль 3. Алгоритм и сложность

- Изучить основные стратегии проектирования алгоритмов, а также различные методы и меры для вычисления алгоритмов
- Знать основные алгоритмы сортировки, используемые при разработке программного обеспечения
- Понять, как различные алгоритмы работают с деревьями, *heaps (кучи)* и графами.
- Понять, как работают *жадные* алгоритмы, их стратегию и примеры их использования в основных известных проблемах. Мы также узнаем о применении жадных алгоритмов на графах
- Мы изучим основные стратегии поиска минимального пути, с приближением существенных проблем данной области и алгоритмов их решения
- Понять технику *Backtracking* и ее основные применения, а также альтернативные техники

Модуль 4. Базы данных

- Изучить различные области применения и назначения систем баз данных, а также их функционирование и архитектуру
- Понимать реляционную модель, от ее структуры и операций до расширенной реляционной алгебры
- Глубоко изучить, что такое базы данных SQL, как они работают, как определять данные и как создавать запросы от самых простых до самых продвинутых и сложных
- Научиться проектировать базы данных с использованием модели "сущность-отношение", создавать диаграммы и узнать о характеристиках расширенной модели E-R
- Углубиться в проектирование реляционных баз данных, анализируя различные нормальные формы и алгоритмы декомпозиции
- Заложить основы для понимания функционирования баз данных NoSQL, а также ознакомиться с базой данных MongoDB

Модуль 5. Расширенные базы данных

- Представить различные системы баз данных, доступные в настоящее время на рынке
- Изучить использование XML и баз данных для работы в Интернете
- Понимать работу расширенных баз данных, таких как параллельные и распределенные базы данных
- Понять важность индексирования и объединения в системах баз данных
- Понимать функционирование систем транзакционной обработки и поиска информации
- Приобрести знания, связанные с нереляционными базами данных и добычей данных

Модуль 6. Продвинутое проектирование алгоритмов

- Углубитесь в продвинутое проектирование алгоритмов, анализируя рекурсивные алгоритмы и алгоритмы "разделяй и властвуй", а также выполняя амортизированный анализ
- Понять концепции динамического программирования и алгоритмы для задач NP
- Понимать, как работает комбинаторная оптимизация, а также различные алгоритмы рандомизации и параллельные алгоритмы
- Знать и понимать, как работают различные методы локального поиска и с кандидатами
- Изучить механизмы формальной проверки программ и итеративной проверки программ, включая логику первого порядка и формальную систему Хоара
- Изучить работу некоторых основных численных методов, таких как метод бисекции, метод Ньютона-Рафсона и метод секущих

Модуль 7. Взаимодействие человека и компьютера

- Приобрести прочные знания в области взаимодействия человека и компьютера и создания удобных интерфейсов
- Понять важность удобства использования приложений и почему важно учитывать его при разработке нашего программного обеспечения
- Понимать различные типы человеческого разнообразия, ограничения, которые они влекут за собой, и как адаптировать интерфейсы в соответствии с конкретными потребностями каждого из них
- Изучить процесс проектирования интерфейса, от анализа требований до оценки, через различные промежуточные этапы, необходимые для реализации подходящего интерфейса
- Знать различные руководящие принципы доступности, стандарты, которые их устанавливают, и инструменты, позволяющие их оценить
- Понять различные методы взаимодействия с компьютером, используя периферийные устройства и приспособления

Модуль 8. Расширенное программирование

- Углубить знания по программированию, особенно в отношении объектно-ориентированного программирования, и различных типов отношений между существующими классами
- Знать различные шаблоны проектирования для решения объектно-ориентированных задач
- Изучить событийно-управляемое программирование и разработку пользовательского интерфейса с помощью Qt
- Приобрести основные знания о параллельном программировании, процессах и потоках
- Научиться управлять использованием потоков и синхронизации, а также решать общие проблемы в рамках параллельного программирования
- Понять важность документации и тестирования при разработке программного обеспечения

Модуль 9. Разработка сетевых приложений

- Знать особенности языка разметки HTML и его использование при создании веб-сайтов вместе с таблицами стилей CSS
- Узнать, как использовать браузерно-ориентированный язык программирования JavaScript и некоторые его основные возможности
- Понять концепции компонентно-ориентированного программирования и компонентной архитектуры
- Узнать, как использовать *Framework* для *Frontend* Bootstrap для создания веб-сайтов
- Понять структуру модели представления контроллера при разработке динамических веб-сайтов
- Знать сервисно-ориентированную архитектуру и основы протокола HTTP

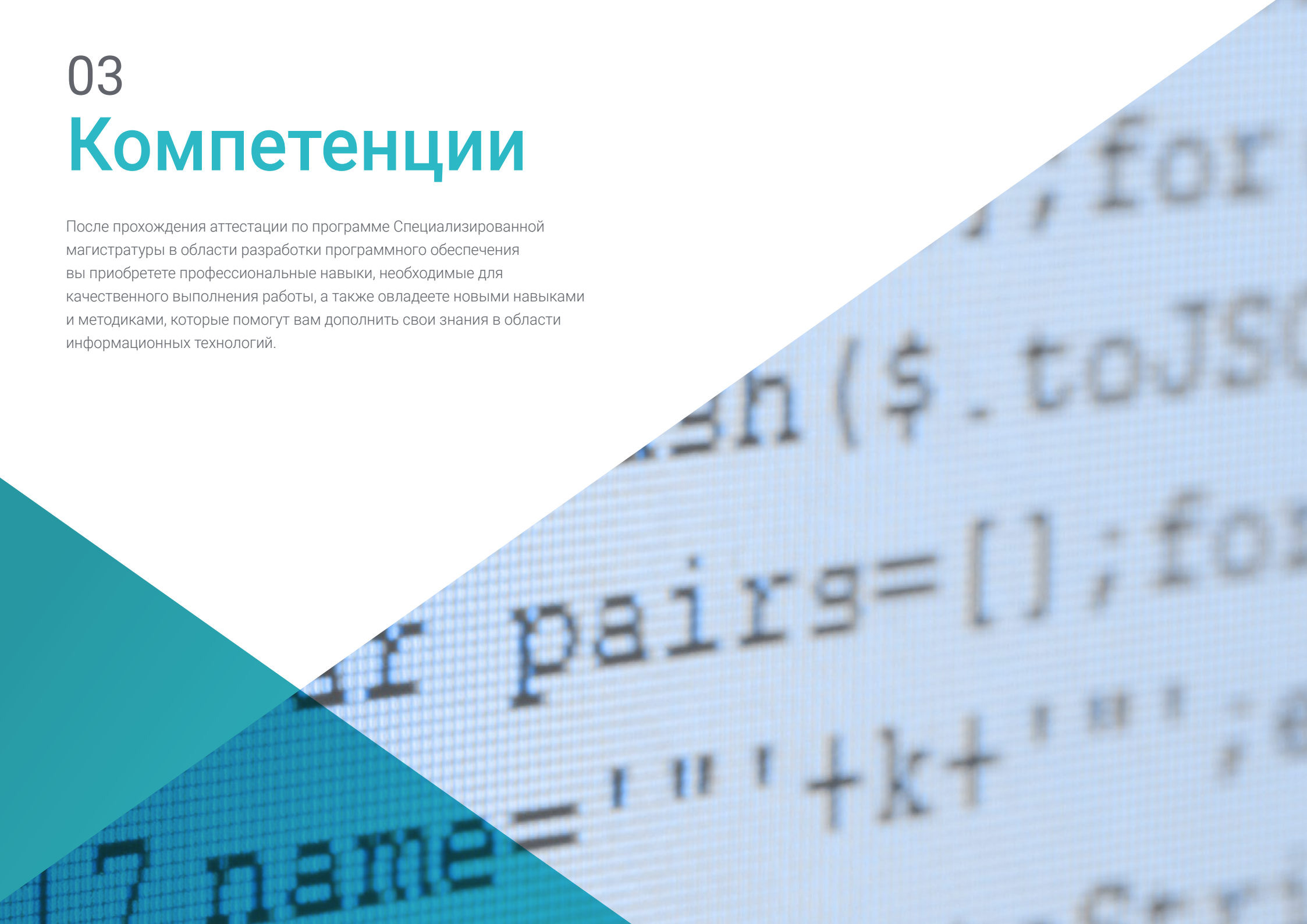
Модуль 10. Программная инженерия

- Заложить основы программной инженерии и моделирования, изучить основные процессы и концепции
- Понимать процесс разработки программного обеспечения и различных моделей разработки программного обеспечения, включая agile-технологии
- Понимать проектирование требований, их разработку, выработку, согласование и проверку
- Изучить моделирование требований и различные элементы, такие как сценарии, информация, классы анализа, поток, поведение и модели
- Понимать концепции и процессы проектирования программного обеспечения, изучение архитектуры проектирования и проектирования на уровне компонентов и паттернов
- Знать основные стандарты, связанные с качеством программного обеспечения и управлением проектами

03

Компетенции

После прохождения аттестации по программе Специализированной магистратуры в области разработки программного обеспечения вы приобретете профессиональные навыки, необходимые для качественного выполнения работы, а также овладеете новыми навыками и методиками, которые помогут вам дополнить свои знания в области информационных технологий.



Повысьте свои навыки в области разработки программного обеспечения и перейдите на новый уровень как профессионал в этой постоянно развивающейся области"

Общий профессиональный навык

» Отвечать на текущие потребности области разработки программного обеспечения

Исключительная программа с точки зрения ее интенсивности, полноты и способа преподавания, которая позволит вам быстро и эффективно развиваться"

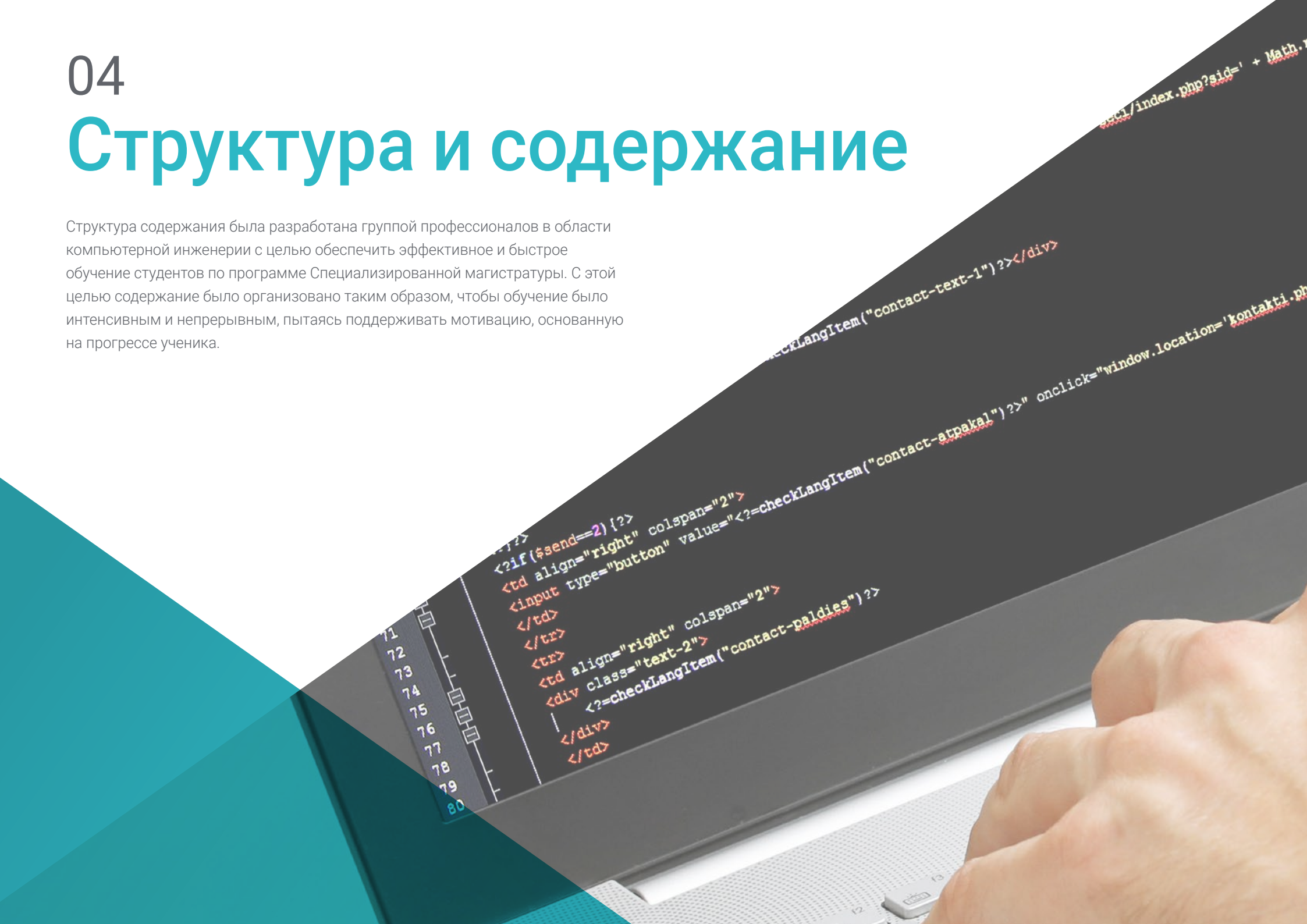
Профессиональные навыки

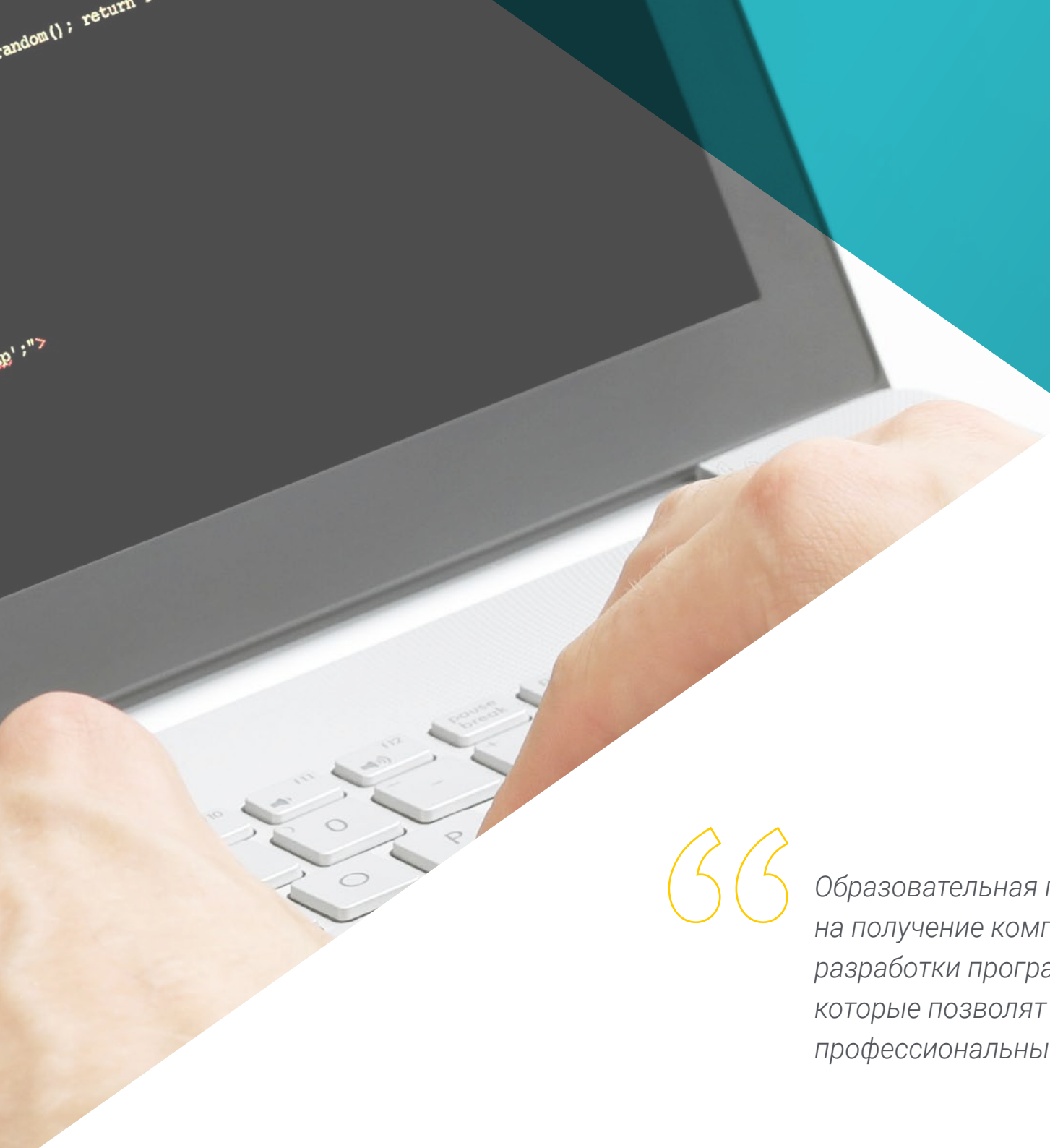
- Уметь понимать базовую структуру компьютера, программное обеспечение и языки программирования общего назначения
- Уметь применять основы программирования на языке C++, включая классы, переменные, условные выражения и объекты
- Углубленно изучить основные стратегии проектирования алгоритмов, а также различные методы и меры для их расчета
- Понимать различные области применения и назначения систем баз данных, а также их работу и архитектуру, и применять их на повседневной основе
- Уметь применять различные системы баз данных, доступные в настоящее время на рынке
- Уметь анализировать рекурсивные алгоритмы и алгоритмы "разделяй и властвуй" и выполнять амортизированный анализ
- Использовать знания о взаимодействии человека и компьютера и создании удобных интерфейсов в повседневной профессиональной деятельности
- Управлять глубокими знаниями программирования
- Знать особенности языка разметки HTML и его использование при создании веб-сайтов вместе с таблицами стилей CSS
- Уметь применять основные процессы и концепции основ программной инженерии и моделирования

04

Структура и содержание

Структура содержания была разработана группой профессионалов в области компьютерной инженерии с целью обеспечить эффективное и быстрое обучение студентов по программе Специализированной магистратуры. С этой целью содержание было организовано таким образом, чтобы обучение было интенсивным и непрерывным, пытаясь поддерживать мотивацию, основанную на прогрессе ученика.





Образовательная программа, направленная на получение комплексных навыков в области разработки программного обеспечения, которые позволят вам выйти на новый профессиональный уровень"

Модуль 1. Основы программирования

1.1. Введение в программирование
- 1.1.1. Базовая структура компьютера
- 1.1.2. Программное обеспечение
- 1.1.3. Языки программирования
- 1.1.4. Жизненный цикл программного приложения

1.2. Проектирование алгоритмов
- 1.2.1. Решение проблем
- 1.2.2. Описательные техники
- 1.2.3. Элементы и структура алгоритма

1.3. Составные части программы
- 1.3.1. Происхождение и характеристики языка C++
- 1.3.2. Среда разработки
- 1.3.3. Концепция программы
- 1.3.4. Основные типы данных
- 1.3.5. Операторы
- 1.3.6. Выражения
- 1.3.7. Утверждения
- 1.3.8. Ввод и вывод данных

1.4. Утверждения контроля
- 1.4.1. Утверждения
- 1.4.2. Бифуркации
- 1.4.3. Циклы

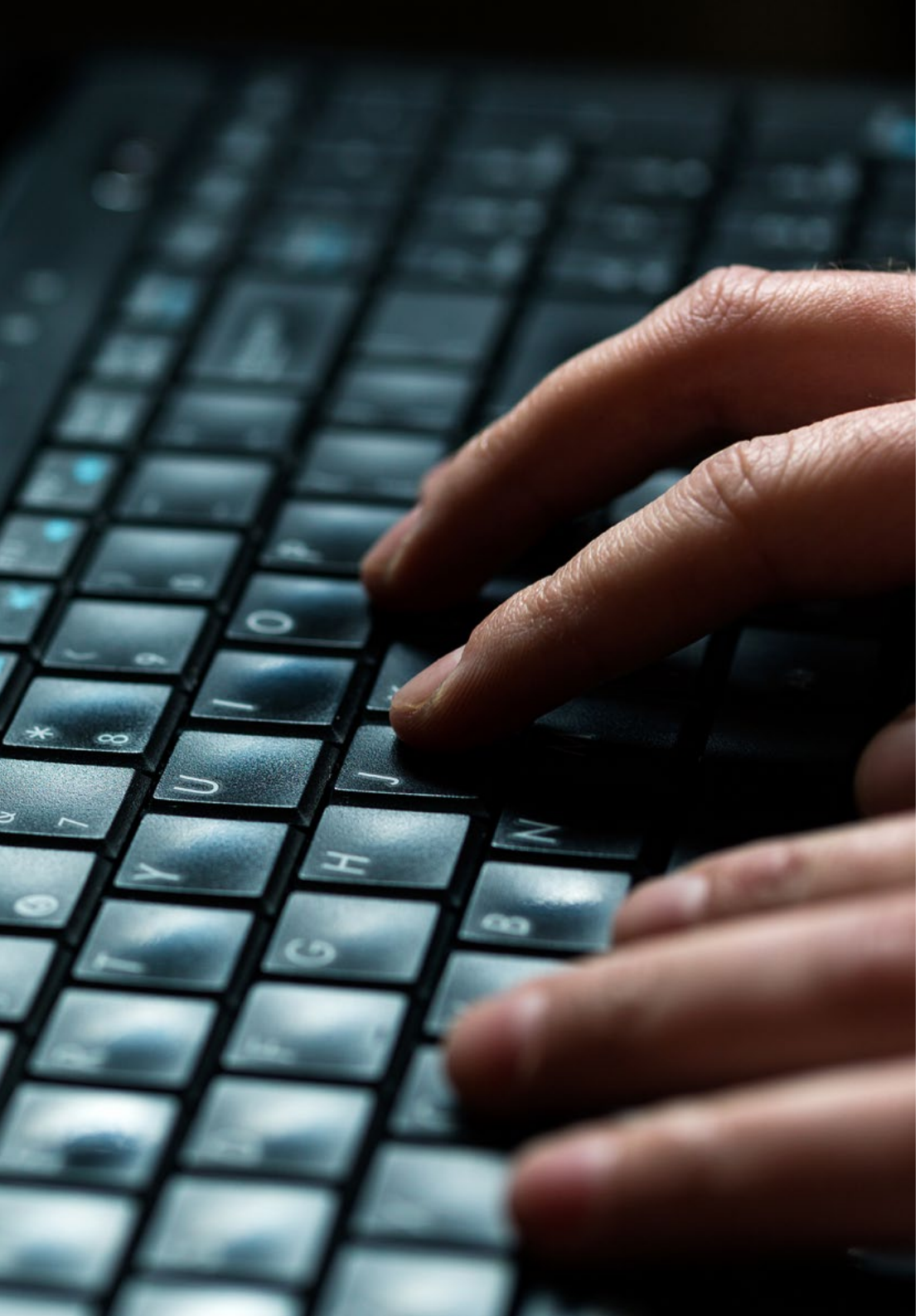
1.5. Абстракция и модульность: функции
- 1.5.1. Модульный дизайн
- 1.5.2. Понятие функции и пользы
- 1.5.3. Определение функции
- 1.5.4. Поток выполнения при вызове функции
- 1.5.5. Прототип функции
- 1.5.6. Возврат результатов
- 1.5.7. Вызов функции: параметры
- 1.5.8. Передача параметров по ссылке и по значению
- 1.5.9. Идентификация области

1.6. Статические структуры данных
- 1.6.1. *Arrays*
- 1.6.2. Матрицы Полиэдры
- 1.6.3. Поиск и сортировка
- 1.6.4. Цепочки. Функции ввода/вывода для строк
- 1.6.5. Структуры. Союзы
- 1.6.6. Новые виды данных

1.7. Динамические структуры данных: указатели
- 1.7.1. Понятие. Определение указателя
- 1.7.2. Операторы и операции с указателями
- 1.7.3. *Arrays* указателей
- 1.7.4. Указатели и *arrays*
- 1.7.5. Указатели на строки
- 1.7.6. Указатели на структуры
- 1.7.7. Множественная косвенность
- 1.7.8. Указатели на функции
- 1.7.9. Передача функций, структур и *arrays* в качестве параметров функции

1.8. Файлы
- 1.8.1. Основные понятия
- 1.8.2. Операции с файлами
- 1.8.3. Типы файлов
- 1.8.4. Организация архивов
- 1.8.5. Введение в файлы C++
- 1.8.6. Управление файлами

1.9. Рекурсивность
- 1.9.1. Определение рекурсии
- 1.9.2. Виды рекурсии
- 1.9.3. Преимущества и недостатки
- 1.9.4. Соображения
- 1.9.5. Итеративная рекурсивная конверсия
- 1.9.6. Стек рекурсии



1.10. Тестирование и документация
- 1.10.1. Тестирование программ
- 1.10.2. Тестирование методом «белого ящика»
- 1.10.3. Тестирование методом «черного ящика»
- 1.10.4. Инструменты для тестирования
- 1.10.5. Программная документация

Модуль 2. Структура данных

2.1 Введение в программирование на C++
- 2.1.1. Классы, конструкторы, методы и атрибуты
- 2.1.2. Переменные
- 2.1.3. Условные выражения и циклы
- 2.1.4. Предметы

2.2. Абстрактные типы данных (ADT)
- 2.2.1. Типы данных
- 2.2.2. Основные структуры и ADT
- 2.2.3. Векторы и *Arrays*

2.3. Линейные структуры данных
- 2.3.1. Определение списка ADT
- 2.3.2. Связанные и дважды связанные списки
- 2.3.3. Упорядоченные списки
- 2.3.4. Списки в C++
- 2.3.5. ADT Стек
- 2.3.6. ADT Очереди
- 2.3.7. Стек и очередь в C++

2.4. Иерархический структуры данных
- 2.4.1. ADT Дерево
- 2.4.2. Маршруты
- 2.4.3. N-арные деревья
- 2.4.4. Бинарные деревья
- 2.4.5. Деревья бинарного поиска

2.5. Иерархические структуры данных: сложные деревья
  2.5.1. Идеально сбалансированные деревья или деревья минимальной высоты
  2.5.2. Многопутевые деревья
  2.5.3. Библиографические ссылки
2.6. Кучи и приоритетная очередь
  2.6.1. Курганы ADT
  2.6.2. ADT Приоритетная очередь
2.7. *Хеш-таблицы*
  2.7.1. *Хеш-таблица* ADT
  2.7.2. *Хеш-функции*
  2.7.3. *Хеш-функция* в *хеш-таблицах*
  2.7.4. Редисперсия
  2.7.5. Открытые *хеш-таблицы*
2.8. Графы
  2.8.1. ADT Графы
  2.8.2. Виды графов
  2.8.3. Графическое представление и основные операции
  2.8.4. Моделирование графов
2.9. Алгоритмы и расширенные концепции графов
  2.9.1. Задачи на графах
  2.9.2. Алгоритмы на путях
  2.9.3. Алгоритмы поиска или пути
  2.9.4. Прочие алгоритмы
2.10. Прочие структуры данных
  2.10.1. Наборы
  2.10.2. *Arrays* параллельные
  2.10.3. Таблица символов
  2.10.4. *Пробы*

Модуль 3. Алгоритм и сложность

3.1. Введение в стратегии проектирования алгоритмов
  3.1.1. Рекурсивность
  3.1.2. Разделяй и властвуй
  3.1.3. Прочие стратегии
3.2. Эффективность и анализ алгоритмов
  3.2.1. Меры эффективности
  3.2.2. Измерение размера входа
  3.2.3. Измерение времени выполнения
  3.2.4. Худший, лучший и средний случай
  3.2.5. Асимптотическое обозначение
  3.2.6. Критерии математического анализа нерекурсивных алгоритмов
  3.2.7. Математический анализ рекурсивных алгоритмов
  3.2.8. Эмпирический анализ алгоритмов
3.3. Алгоритмы сортировки
  3.3.1. Концепция организации
  3.3.2. Сортировка пузырьком
  3.3.3. Сортировка по выбору
  3.3.4. Сортировка вставками
  3.3.5. Сортировка слиянием (Merge Sort)
  3.3.6. Быстрая сортировка (QuickSort)
3.4. Алгоритмы на деревьях
  3.4.1. Концепция деревьев
  3.4.2. Бинарные деревья
  3.4.3. Обход дерева
  3.4.4. Представление выражений
  3.4.5. Упорядоченные бинарные деревья
  3.4.6. Сбалансированные бинарные деревья
3.5. Алгоритмы с *Heaps (кучи)*
  3.5.1. *Heaps (кучи)*
  3.5.2. Алгоритм HeapSort
  3.5.3. Приоритетные очереди

3.6. Алгоритмы с применением графов
- 3.6.1. Представление
- 3.6.2. Обход по ширине
- 3.6.3. Углубленный обход
- 3.6.4. Топологическая сортировка

3.7. *Жадные* алгоритмы
- 3.7.1. *Жадная* стратегия
- 3.7.2. Элементы *жадной* стратегии
- 3.7.3. Обмен валюты
- 3.7.4. Задача коммивояжёра
- 3.7.5. Задача о рюкзаке

3.8. Поиск минимальных путей
- 3.8.1. Проблема минимального пути
- 3.8.2. Отрицательные дуги и циклы
- 3.8.3. Алгоритм Дейкстры

3.9. *Жадные* алгоритмы на графах
- 3.9.1. Дерево минимального покрытия
- 3.9.2. Алгоритм Прима
- 3.9.3. Алгоритм Крускала
- 3.9.4. Анализ сложности

3.10. *Возврат*
- 3.10.1. *Backtracking*
- 3.10.2. Альтернативные методы

Модуль 4. Базы данных

4.1. Применение и назначение систем баз данных
- 4.1.1. Применение различных систем баз данных
- 4.1.2. Назначение в различных системах баз данных
- 4.1.3. Обзор данных

4.2. База данных и архитектура
- 4.2.1. Реляционные базы данных
- 4.2.2. Проектирование базы данных
- 4.2.3. Объектно-ориентированные и полуструктурированные базы данных
- 4.2.4. Хранение и запросы данных
- 4.2.5. Управление транзакциями
- 4.2.6. Добыча и анализ данных
- 4.2.7. Архитектура базы данных

4.3. Реляционная модель: структура, операции и расширенная реляционная алгебра
- 4.3.1. Структура реляционных баз данных
- 4.3.2. Фундаментальные операции в реляционной алгебре
- 4.3.3. Прочие операции реляционной алгебры
- 4.3.4. Расширенные операции реляционной алгебры
- 4.3.5. Нулевые значения
- 4.3.6. Внесение изменений в базу данных

4.4. SQL (I)
- 4.4.1. Что такое SQL?
- 4.4.2. Определение данных
- 4.4.3. Базовая структура SQL-запросов
- 4.4.4. Операции над наборами
- 4.4.5. Агрегатные функции
- 4.4.6. Нулевые значения

4.5. SQL (II)
- 4.5.1. Вложенные запросы
- 4.5.2. Сложные запросы
- 4.5.3. Просмотры
- 4.5.4. Курсоры
- 4.5.5. Сложные запросы
- 4.5.6. Триггеры

4.6. Проектирование баз данных и ER-модель
- 4.6.1. Обзор процесса проектирования
- 4.6.2. Модель сущности-отношения
- 4.6.3. Ограничения

4.7. Диаграммы сущность-связь
- 4.7.1. Диаграммы сущность-связь
- 4.7.2. Свойства проектирования сущность-отношение
- 4.7.3. Наборы слабых сущностей

4.8. Расширенная модель сущность-отношение
- 4.8.1. Характеристики расширенной ER-модели
- 4.8.2. Проектирование базы данных
- 4.8.3. Сокращение до реляционных схем

4.9. Разработка реляционной базы данных
- 4.9.1. Характеристики хороших реляционных проектов
- 4.9.2. Атомные домены и первая нормальная форма (1FN)
- 4.9.3. Декомпозиция через функциональные зависимости
- 4.9.4. Теория функциональной зависимости
- 4.9.5. Алгоритмы декомпозиции
- 4.9.6. Декомпозиция через многозначные зависимости
- 4.9.7. Другие нормальные формы
- 4.9.8. Процесс разработки базы данных

4.10. Базы данных NoSQL
- 4.10.1. Что представляют собой базы данных NoSQL?
- 4.10.2. Анализ различных вариантов NoSQL и их особенностей
- 4.10.3. MongoDB

Модуль 5. Расширенные базы данных

5.1. Введение в различные системы баз данных
- 5.1.1. Исторический обзор
- 5.1.2. Иерархические базы данных
- 5.1.3. Сетевые базы данных
- 5.1.4. Реляционные базы данных
- 5.1.5. Нереляционные базы данных

5.2. XML и базы данных для веб-сайта
- 5.2.1. Валидация XML-файлов
- 5.2.2. Преобразования XML-файлов
- 5.2.3. Хранение данных в XML
- 5.2.4. Реляционные базы данных XML
- 5.2.5. SQL/XML
- 5.2.6. Собственные базы данных XML

5.3. Параллельные базы данных
- 5.3.1. Параллельные системы
- 5.3.2. Параллельные архитектуры баз данных
- 5.3.4. Параллелизм в запросах
- 5.3.5. Параллелизм при запросах
- 5.3.6. Проектирование параллельных систем
- 5.3.7. Параллельная обработка в SQL

5.4. Распределенные базы данных
- 5.4.1. Распределенные системы
- 5.4.2. Распределенное хранение
- 5.4.3. Доступность
- 5.4.4. Распределенная обработка запросов
- 5.4.5. Провайдеры распределенных баз данных

5.5. Индексация и ассоциация
- 5.5.1. Упорядоченные индексы
- 5.5.2. Разреженные и плотные индексы
- 5.5.3. Многоуровневые индексы
- 5.5.4. Обновление индекса
- 5.5.5. Статическая ассоциация
- 5.5.6. Как применять индексы в базах данных

5.6. Введение в транзакционную обработку
- 5.6.1. Стадии транзакции
- 5.6.2. Внедрение атомарности и долговечности
- 5.6.3. Последовательность
- 5.6.4. Восстановление
- 5.6.5. Внедрение изоляции

5.7. Системы восстановления
- 5.7.1. Классификация ошибок
- 5.7.2. Структуры хранения
- 5.7.3. Восстановление и атомарность
- 5.7.4. Восстановление на основе исторических данных
- 5.7.5. Параллельные транзакции и восстановление
- 5.7.6. Высокая доступность в базах данных

5.8. Выполнение и обработка запросов
- 5.8.1. Стоимость консультации
- 5.8.2. Работа по выбору
- 5.8.3. Ординация
- 5.8.4. Введение в оптимизацию запросов
- 5.8.5. Мониторинг эффективности

5.9. Нереляционные базы данных
- 5.9.1. Документо-ориентированные базы данных
- 5.9.2. Графовые базы данных
- 5.9.3. Базы данных с ключами-значениями

5.10. *Хранилище* данных, OLAP и добыча данных
- 5.10.1. Компоненты хранилищ данных
- 5.10.2. Архитектура *хранилища* данных
- 5.10.3. OLAP
- 5.10.4. Возможности добычи данных
- 5.10.5. Другие виды добычи данных

Модуль 6. Продвинутое проектирование алгоритмов

6.1. Анализ рекурсивных алгоритмов и алгоритмов «разделяй и властвуй»
- 6.1.1. Составление и решение однородных и неоднородных рекуррентных уравнений
- 6.1.2. Суть метода разделяй и властвуй

6.2. Амортизированный анализ
- 6.2.1. Агрегатный анализ
- 6.2.2. Метод учета
- 6.2.3. Метод потенциалов

6.3. Динамическое программирование и алгоритмы для NP-задач
- 6.3.1. Характеристики динамического программирования
- 6.3.2. Обратный путь: *Backtracking*
- 6.3.3. Ветвление и подрезка

6.4. Комбинаторная оптимизация
- 6.4.1. Представление проблем
- 6.4.2. 1D оптимизация

6.5. Рандомизированные алгоритмы
- 6.5.1. Примеры алгоритмов рандомизации
- 6.5.2. Теорема Бюффона
- 6.5.3. Алгоритм Монте-Карло
- 6.5.4. Алгоритм Лас-Вегаса

6.6. Локальный поиск и с кандидатов
- 6.6.1. *Градиентный подъем*
- 6.6.2. *Поиск восхождением к вершине*
- 6.6.3. *Имитационный отжиг*
- 6.6.4. *Поиск с запретами*
- 6.6.5. Поиск с кандидатами

6.7. Формальная верификация программ
- 6.7.1. Определение функциональных абстракций
- 6.7.2. Язык логики первого порядка
- 6.7.3. Формальная система Хоара

6.8. Верификация итеративных программ
- 6.8.1. Правила формальной системы Хоара
- 6.8.2. Концепция инвариантных итераций

6.9. Численные методы
- 6.9.1. Метод бисекции
- 6.9.2. Метод Ньютона-Рафсона
- 6.9.3. Метод секущей

6.10. Параллельные алгоритмы
- 6.10.1. Параллельные бинарные операции
- 6.10.2. Параллельные вычисления на графах
- 6.10.3. Параллелизм в «разделяй и властвуй»
- 6.10.4. Параллелизм в динамическом программировании

Модуль 7. Взаимодействие человека и компьютера

7.1. Введение в взаимодействие человека и компьютера
- 7.1.1. Что такое взаимодействие человека и компьютера?
- 7.1.2. Взаимодействия человека и компьютера с другими дисциплинами
- 7.1.3. Пользовательский интерфейс
- 7.1.4. Удобство использования и доступность
- 7.1.5. Опыт пользователей и дизайна, ориентированного на пользователя

7.2. Компьютер и взаимодействие: пользовательский интерфейс и парадигмы взаимодействия
- 7.2.1. Взаимодействие
- 7.2.2. Парадигмы и стили взаимодействия
- 7.2.3. Эволюция пользовательских интерфейсов
- 7.2.4. Классические пользовательские интерфейсы: WIMP/GUI, команды, голос, виртуальная реальность
- 7.2.5. Инновационные пользовательские интерфейсы: мобильные, носимые, совместные, НКИ

7.3. Человеческий фактор: психологические и когнитивные аспекты
- 7.3.1. Важность человеческого фактора, основанная на взаимодействии
- 7.3.2. Обработка информации человеком
- 7.3.3. Ввод и вывод информации: визуальной, слуховой и тактильной
- 7.3.4. Восприятие и внимание
- 7.3.5. Знания и ментальные модели: представление, организация и приобретение

7.4. Человеческий фактор: сенсорные и физические ограничения
- 7.4.1. Функциональное разнообразие, инвалидность и ограничения жизнедеятельности
- 7.4.2. Визуальное разнообразие
- 7.4.3. Слуховое разнообразие
- 7.4.4. Когнитивное разнообразие
- 7.4.5. Моторно-функциональное разнообразие
- 7.4.6. Случай цифровых мигрантов

7. 5. Процесс проектирования (I): анализ требований для проектирования пользовательского интерфейса
   - 7.5.1. Дизайн, ориентированный на пользователя
   - 7.5.2. Что такое анализ требований
   - 7.5.3. Сбор информации
   - 7.5.4. Анализ и интерпретация информации
   - 7.5.5. Анализ юзабилити и доступность
7.6. Процесс проектирования (II): создание прототипов и анализ задач
   - 7.6.1. Концептуальное проектирование
   - 7.6.2. Создание прототипов
   - 7.6.3. Иерархический анализ задач
7.7. Процесс проектирования (III): оценка
   - 7.7.1. Оценка в процессе проектирования: цели и методы
   - 7.7.2. Методы оценки без участия пользователей
   - 7.7.3. Методы оценки с участием пользователей
   - 7.7.4. Стандарты и нормы оценки
7.8. Доступность: определение и рекомендации
   - 7.8.1. Доступность и универсальный дизайн
   - 7.8.2. Инициатива WAI и руководство WCAG
   - 7.8.3. Руководство WCAG 2.0 и 2.1
7.9. Доступность: оценка и функциональное разнообразие
   - 7.9.1. Инструменты оценки доступности веб-сайтов
   - 7.9.2. Доступность и функциональное разнообразие
7.10. Компьютер и взаимодействие: периферийные устройства и приспособления
   - 7.10.1. Традиционные периферийные устройства
   - 7.10.2. Альтернативные периферийные устройства
   - 7.10.3. Мобильные телефоны и планшеты
   - 7.10.4. Функциональное разнообразие, взаимодействие и периферийные устройства

Модуль 8. Расширенное программирование

8.1. Введение в объектно-ориентированное программирование
   - 8.1.1. Введение в объектно-ориентированное программирование
   - 8.1.2. Проектирование классов
   - 8.1.3. Введение в UML для моделирования проблем
8.2. Отношения между классами
   - 8.2.1. Абстракция и наследство
   - 8.2.2. Расширенные концепции наследования
   - 8.2.3. Полиморфизм
   - 8.2.4. Состав и агрегация
8.3. Введение в шаблоны проектирования для решения объектно-ориентированных задач
   - 8.3.1. Что такое шаблоны проектирования?
   - 8.3.2. Шаблон *Factory*
   - 8.3.4. Шаблон *Singleton*
   - 8.3.5. Шаблон *Observer*
   - 8.3.6. Шаблон *Composite*
8.4. Исключения
   - 8.4.1. Что такое исключения?
   - 8.4.2. Перехват и обработка исключений
   - 8.4.3. Запуск исключений
   - 8.4.4. Создание исключений
8.5. Пользовательские интерфейсы
   - 8.5.1. Введение в Qt
   - 8.5.2. Позиционирование
   - 8.5.3. Что такое события?
   - 8.5.4. События: определение и фиксация
   - 8.5.5. Разработка пользовательского интерфейса
8.6. Введение в параллельное программирование
   - 8.6.1. Введение в параллельное программирование
   - 8.6.2. Понятие процесса и потока
   - 8.6.3. Взаимодействие между процессами или потоками
   - 8.6.4. Потоки в C++

8.6.6. Преимущества и недостатки параллельного программирования
8.7. Управление и синхронизация потоков
8.7.1. Жизненный цикл потока
8.7.2. Класс *Thread*
8.7.3. Планирование потоков
8.7.4. Группы потоков
8.7.5. Потоки демонического типа
8.7.6. Синхронизация
8.7.7. Механизмы блокировки
8.7.8. Механизмы коммуникации
8.7.9. Мониторы
8.8. Распространенные проблемы параллельного программирования
8.8.1. Задача о производителях и потребителях
8.8.2. Задача о читателях-писателях
8.8.3. Задача об обедающих философах
8.9. Документация и тестирование ПО
8.9.1. Почему важно документировать ПО?
8.9.2. Проектный документ
8.9.3. Использование инструментов для документирования
8.10. Тестирование ПО
8.10.1. Введение в тестирование ПО
8.10.2. Виды тестирования
8.10.3. Модульный тест
8.10.4. Интеграционный тест
8.10.5. Валидационный тест
8.10.6. Системный тест

Модуль 9. Разработка сетевых приложений

9.1. Языки разметки HTML5
9.1.1. Основные концепции HTML
9.1.2. Новые элементы HTML 5
9.1.3. Формы: новые элементы управления
9.2. Введение в таблицы стилей CSS
9.2.1. Первые шаги с CSS
9.2.2. Введение в CSS3
9.3. Браузерные скрипты: JavaScript
9.3.1. Основные концепции JavaScript
9.3.2. DOM
9.3.3. События
9.3.4. JQuery
9.3.5. Ajax
9.4. Концепция компонентно-ориентированного программирования
9.4.1. Контекст
9.4.2. Компоненты и интерфейсы
9.4.3. Состояния компонента
9.5. Архитектура компонентов
9.5.1. Текущие архитектуры
9.5.2. Интеграция и развертывание компонентов
9.6. *Framework Frontend*: Bootstrap
9.6.1. Проектирование сетки
9.6.2. Формуляры
9.6.3. Компоненты
9.7. Контроллер представления модели
9.7.1. Методы веб-разработки
9.7.2. Шаблон проектирования: MVC
9.8. Информационные грид-технологии
9.8.1. Увеличение вычислительных ресурсов
9.8.2. Концепция грид-технологии

9.9. Сервис-ориентированная архитектура
- 9.9.1. SOA и веб-сервисы
- 9.9.2. Топология веб-сервиса
- 9.9.3. Платформы для веб-сервисов

9.10. Протокол HTTP
- 9.10.1. Сообщения
- 9.10.2. Постоянные сеансы
- 9.10.3. Криптографическая система
- 9.10.4. Принцип работы протокола HTTPS

Модуль 10. Программная инженерия

10.1. Введение в программную инженерию и моделирование
- 10.1.1. Природа программного обеспечения
- 10.1.2. Уникальная природа WebApps
- 10.1.3. Программная инженерия
- 10.1.4. Программный процесс
- 10.1.5. Практика программной инженерии
- 10.1.6. Мифы программного обеспечения
- 10.1.7. Как все начинается
- 10.1.8. Объектно-ориентированные концепции
- 10.1.9. Введение в UML

10.2. Программный процесс
- 10.2.1. Общая модель процесса
- 10.2.2. Предписывающие модели процессов
- 10.2.3. Специализированные модели процессов
- 10.2.4. Унифицированный процесс
- 10.2.5. Модели персональных и командных процессов
- 10.2.6. Что такое оперативность?
- 10.2.7. Что такое agile-процесс?
- 10.2.8. Scrum
- 10.2.9. Набор инструментов agile-процессов

10.3. Принципы, определяющие практику программной инженерии
- 10.3.1. Принципы, которыми руководствуются в процессе
- 10.3.2. Принципы, которыми руководствуется практика
- 10.3.3. Принципы коммуникации
- 10.3.4. Принципы планирования
- 10.3.5. Принципы моделирования
- 10.3.6. Принципы построения
- 10.3.7. Принципы развертывания

10.4. Понимание требований
- 10.4.1. Инженерия требований
- 10.4.2. Установление основ
- 10.4.3. Изучение требований
- 10.4.4. Разработка сценариев использования
- 10.4.5. Создание модели требований
- 10.4.6. Согласование требований
- 10.4.7. Валидация требований

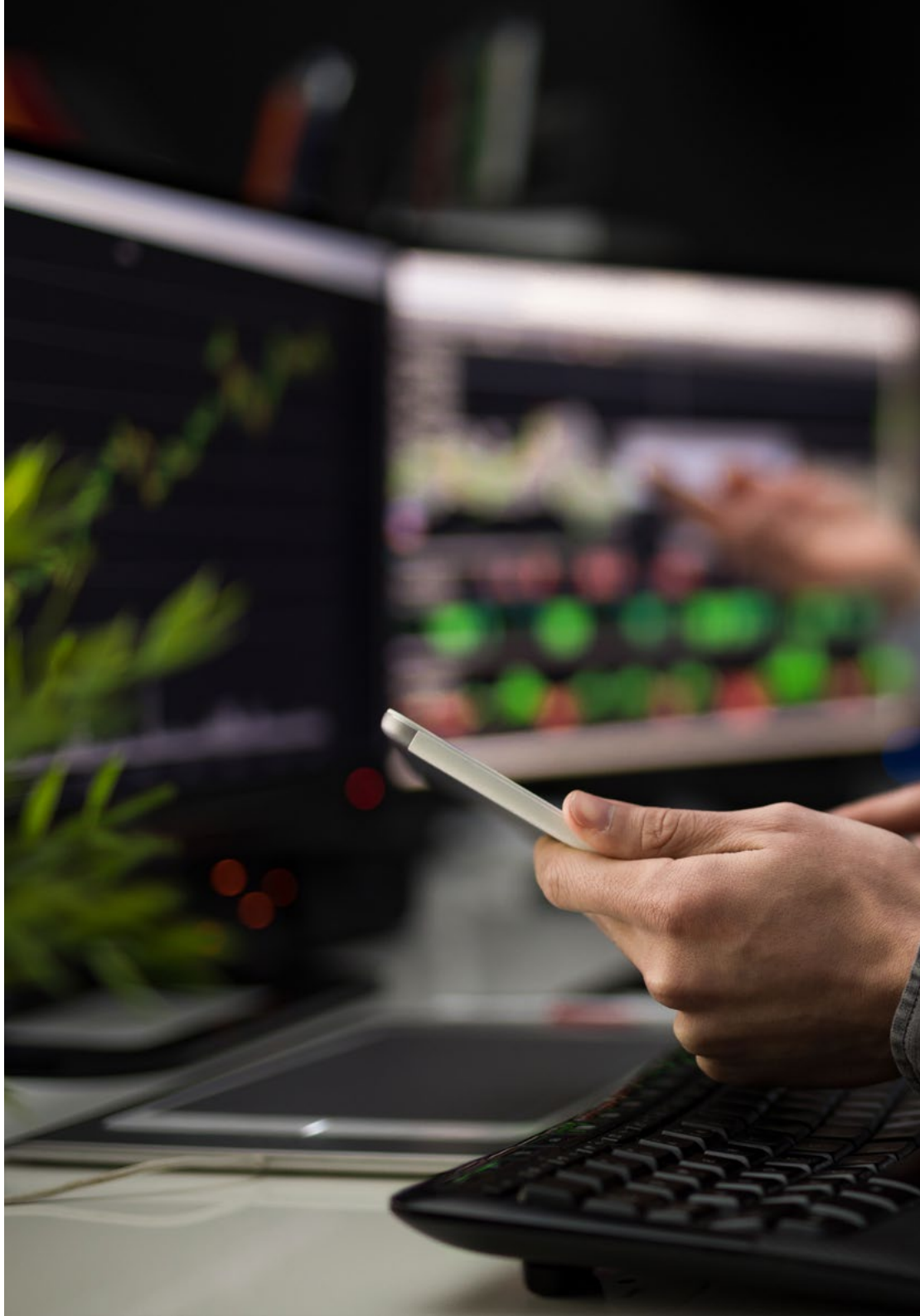
10.5. Моделирование требований: сценарии, информация и классы анализа
- 10.5.1. Анализ требований
- 10.5.2. Моделирование, основанное на сценариях
- 10.5.3. UML-модели, обеспечивающие вариант использования
- 10.5.4. Концепции моделирования данных
- 10.5.5. Моделирование, основанное на классах
- 10.5.6. Диаграммы классов

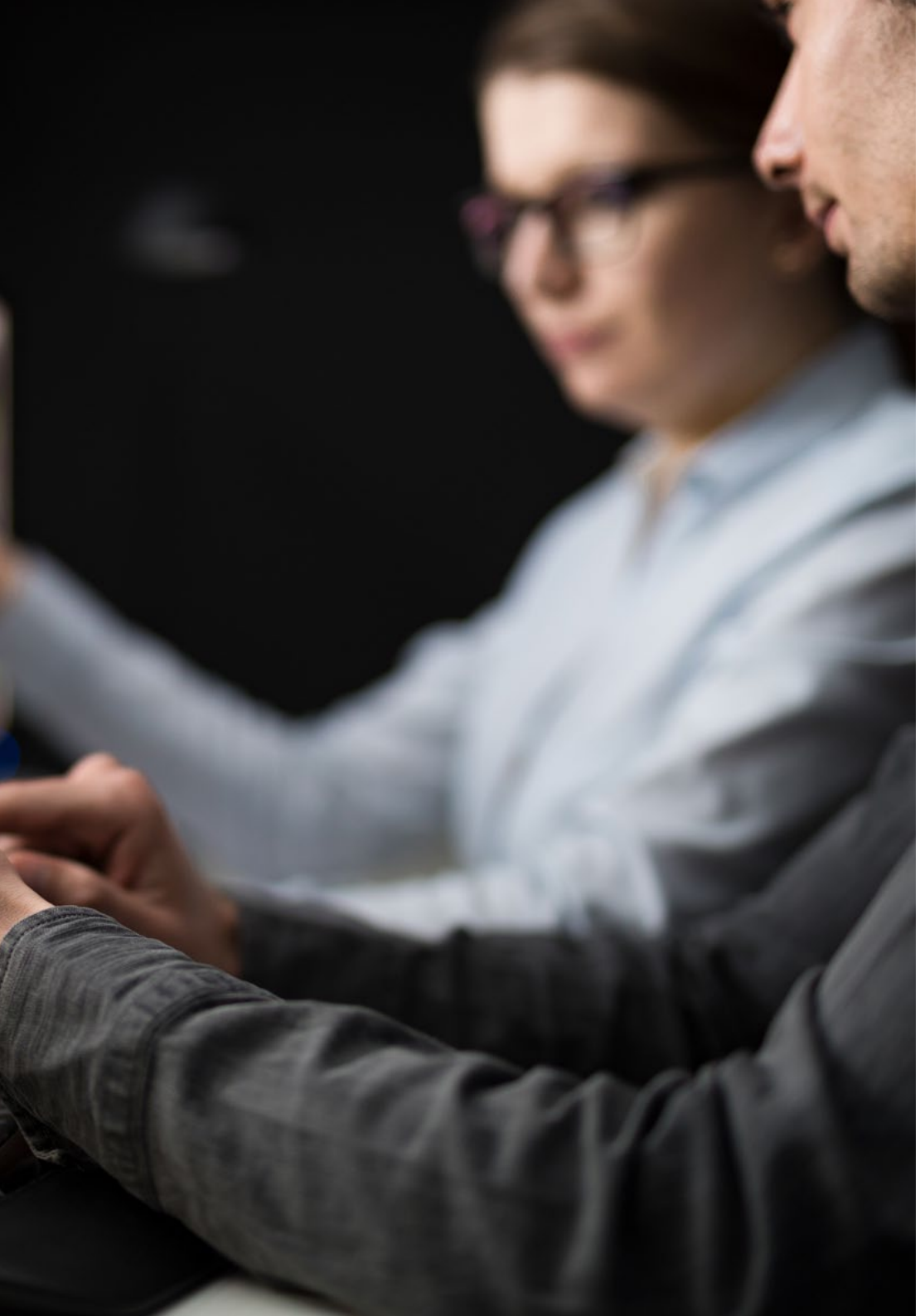
10.6. Моделирование требований: поток, поведение и паттерны
- 10.6.1. Стратегии формирования требований
- 10.6.2. Моделирование, ориентированное на поток
- 10.6.3. Диаграммы состояний
- 10.6.4. Создание модели поведения
- 10.6.5. Диаграммы последовательности
- 10.6.6. Диаграммы коммуникаций
- 10.6.7. Шаблоны для проектирования требований

10.7. Концепции проектирования
- 10.7.1. Проектирование в контексте программной инженерии
- 10.7.2. Процесс дизайна
- 10.7.3. Концепции проектирования
- 10.7.4. Концепции объектно-ориентированного проектирования
- 10.7.5. Модель проектирования

10.8. Архитектурное проектирование
- 10.8.1. Архитектура программного обеспечения
- 10.8.2. Архитектурные жанры
- 10.8.3. Архитектурные стили
- 10.8.4. Архитектурное проектирование
- 10.8.5. Эволюция альтернативных проектов архитектуры
- 10.8.6. Построение карты архитектуры с использованием потоков данных

10.9. Проектирование на компонентном уровне и на основе паттернов
- 10.9.1. Что такое компонент?
- 10.9.2. Проектирование компонентов на основе классов
- 10.9.3. Реализация проекта на уровне компонентов
- 10.9.4. Проектирование традиционных компонентов
- 10.9.5. Компонентно-ориентированная разработка
- 10.9.6. Паттерны проектирования
- 10.9.7. Проектирование программного обеспечения на основе паттернов
- 10.9.8. Архитектурные паттерны
- 10.9.9. Паттерны проектирования на уровне компонентов
- 10.9.10. Паттерны проектирования пользовательского интерфейса





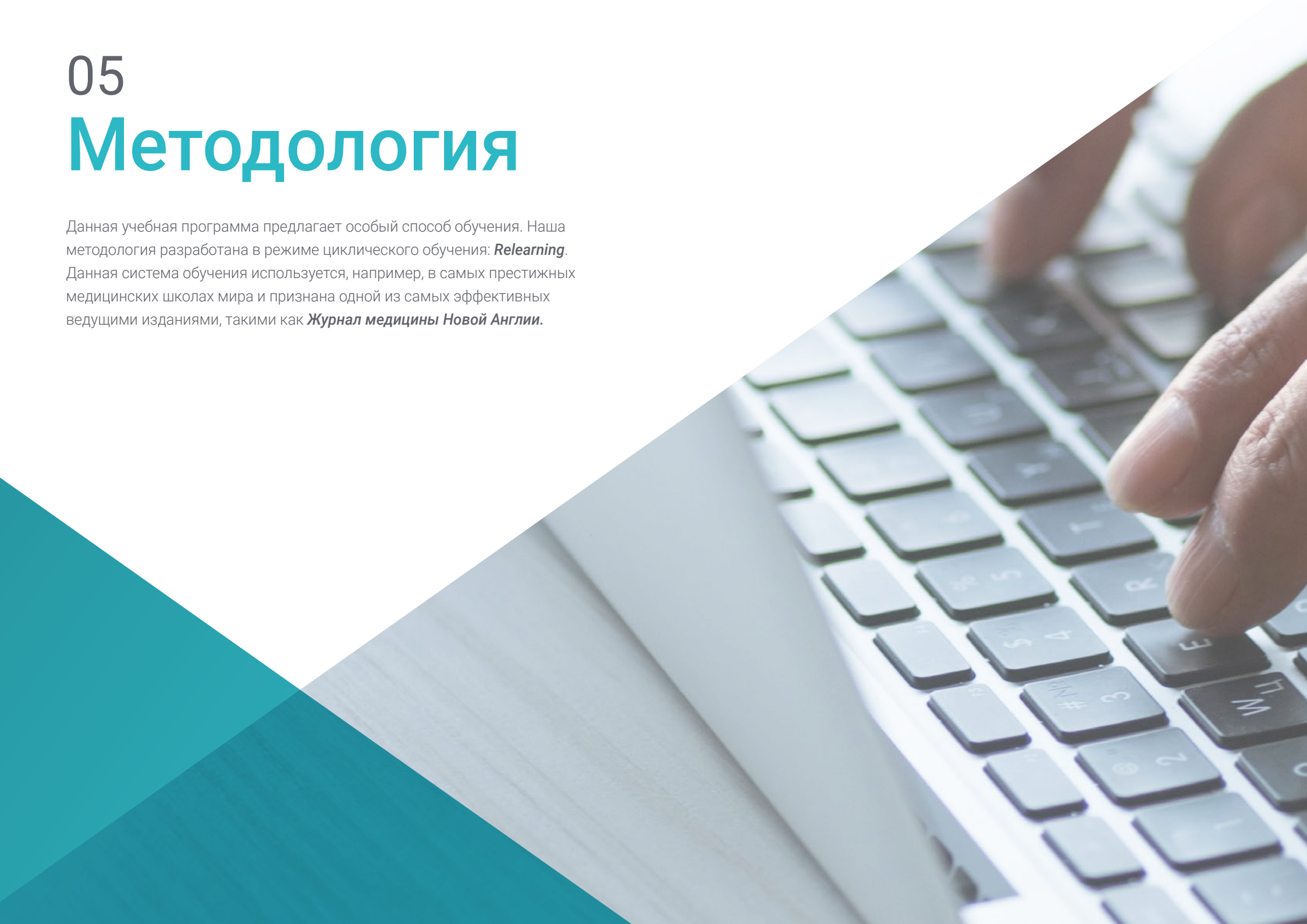
10.10. Качество ПО и управление проектами
- 10.10.1. Качество
- 10.10.1. Качество ПО
- 10.10.2. Дилемма качества ПО
- 10.10.3. Достижение качества ПО
- 10.10.4. Проверка качества ПО
- 10.10.5. Административный спектр
- 10.10.6. Персонал
- 10.10.7. Продукт
- 10.10.8. Процесс
- 10.10.9. Проект
- 10.10.10. Принципы и практика

Уникальный, важный и значимый курс обучения для повышения вашей квалификации"

05

Методология

Данная учебная программа предлагает особый способ обучения. Наша методология разработана в режиме циклического обучения: ***Relearning***. Данная система обучения используется, например, в самых престижных медицинских школах мира и признана одной из самых эффективных ведущими изданиями, такими как ***Журнал медицины Новой Англии.***



Откройте для себя методику Relearning, которая отвергает традиционное линейное обучение, чтобы показать вам циклические системы обучения: способ, который доказал свою огромную эффективность, особенно в предметах, требующих запоминания"

Исследование кейсов для контекстуализации всего содержания

Наша программа предлагает революционный метод развития навыков и знаний. Наша цель - укрепить компетенции в условиях меняющейся среды, конкуренции и высоких требований.

С TECH вы сможете познакомиться со способом обучения, который опровергает основы традиционных методов образования в университетах по всему миру”

Вы получите доступ к системе обучения, основанной на повторении, с естественным и прогрессивным обучением по всему учебному плану.

В ходе совместной деятельности и рассмотрения реальных кейсов студент научится разрешать сложные ситуации в реальной бизнес-среде.

Инновационный и отличный от других метод обучения

Эта программа TECH - интенсивная программа обучения, созданная с нуля, которая предлагает самые сложные задачи и решения в этой области на международном уровне. Благодаря этой методологии ускоряется личностный и профессиональный рост, делая решающий шаг на пути к успеху. Метод кейсов, составляющий основу данного содержания, обеспечивает следование самым современным экономическим, социальным и профессиональным реалиям.

> *Наша программа готовит вас к решению новых задач в условиях неопределенности и достижению успеха в карьере"*

Кейс-метод является наиболее широко используемой системой обучения лучшими преподавателями в мире. Разработанный в 1912 году для того, чтобы студенты-юристы могли изучать право не только на основе теоретического содержания, метод кейсов заключается в том, что им представляются реальные сложные ситуации для принятия обоснованных решений и ценностных суждений о том, как их разрешить. В 1924 году он был установлен в качестве стандартного метода обучения в Гарвардском университете.

Что должен делать профессионал в определенной ситуации? Именно с этим вопросом мы сталкиваемся при использовании кейс-метода - метода обучения, ориентированного на действие. На протяжении всей курса студенты будут сталкиваться с многочисленными реальными случаями из жизни. Им придется интегрировать все свои знания, исследовать, аргументировать и защищать свои идеи и решения.

Методология *Relearning*

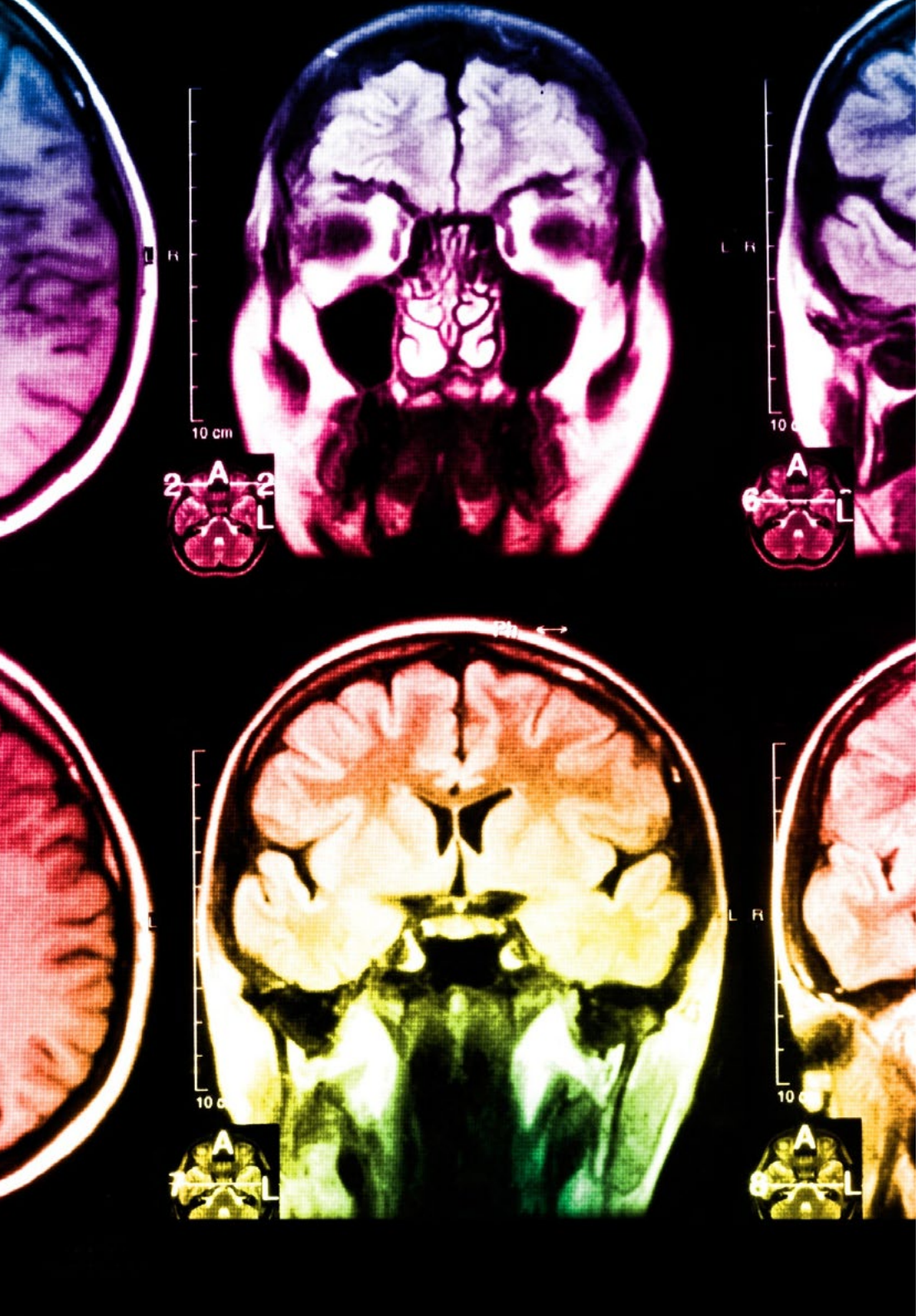
TECH эффективно объединяет метод кейсов с системой 100% онлайн-обучения, основанной на повторении, которая сочетает различные дидактические элементы в каждом уроке.

Мы улучшаем метод кейсов с помощью лучшего метода 100% онлайн-обучения: *Relearning*.

В 2019 году мы достигли лучших результатов обучения среди всех онлайн-университетов в мире.

В TECH вы будете учиться по передовой методике, разработанной для подготовки руководителей будущего. Этот метод, играющий ведущую роль в мировой педагогике, называется *Relearning*.

Наш университет - единственный вуз, имеющий лицензию на использование этого успешного метода. В 2019 году нам удалось повысить общий уровень удовлетворенности наших студентов (качество преподавания, качество материалов, структура курса, цели...) по отношению к показателям лучшего онлайн-университета.



В нашей программе обучение не является линейным процессом, а происходит по спирали (мы учимся, разучиваемся, забываем и заново учимся). Поэтому мы дополняем каждый из этих элементов по концентрическому принципу. Благодаря этой методике более 650 000 выпускников университетов добились беспрецедентного успеха в таких разных областях, как биохимия, генетика, хирургия, международное право, управленческие навыки, спортивная наука, философия, право, инженерное дело, журналистика, история, финансовые рынки и инструменты. Наша методология преподавания разработана в среде с высокими требованиями к уровню подготовки, с университетским контингентом студентов с высоким социально-экономическим уровнем и средним возрастом 43,5 года.

Методика Relearning позволит вам учиться с меньшими усилиями и большей эффективностью, все больше вовлекая вас в процесс обучения, развивая критическое мышление, отстаивая аргументы и противопоставляя мнения, что непосредственно приведет к успеху.

Согласно последним научным данным в области нейронауки, мы не только знаем, как организовать информацию, идеи, образы и воспоминания, но и знаем, что место и контекст, в котором мы что-то узнали, имеют фундаментальное значение для нашей способности запомнить это и сохранить в гиппокампе, чтобы удержать в долгосрочной памяти.

Таким образом, в рамках так называемого нейрокогнитивного контекстно-зависимого электронного обучения, различные элементы нашей программы связаны с контекстом, в котором участник развивает свою профессиональную практику.

В рамках этой программы вы получаете доступ к лучшим учебным материалам, подготовленным специально для вас:

Учебный материал

Все дидактические материалы создаются преподавателями специально для студентов этого курса, чтобы они были действительно четко сформулированными и полезными.

Затем вся информация переводится в аудиовизуальный формат, создавая дистанционный рабочий метод TECH. Все это осуществляется с применением новейших технологий, обеспечивающих высокое качество каждого из представленных материалов.

Мастер-классы

Существуют научные данные о пользе экспертного наблюдения третьей стороны.

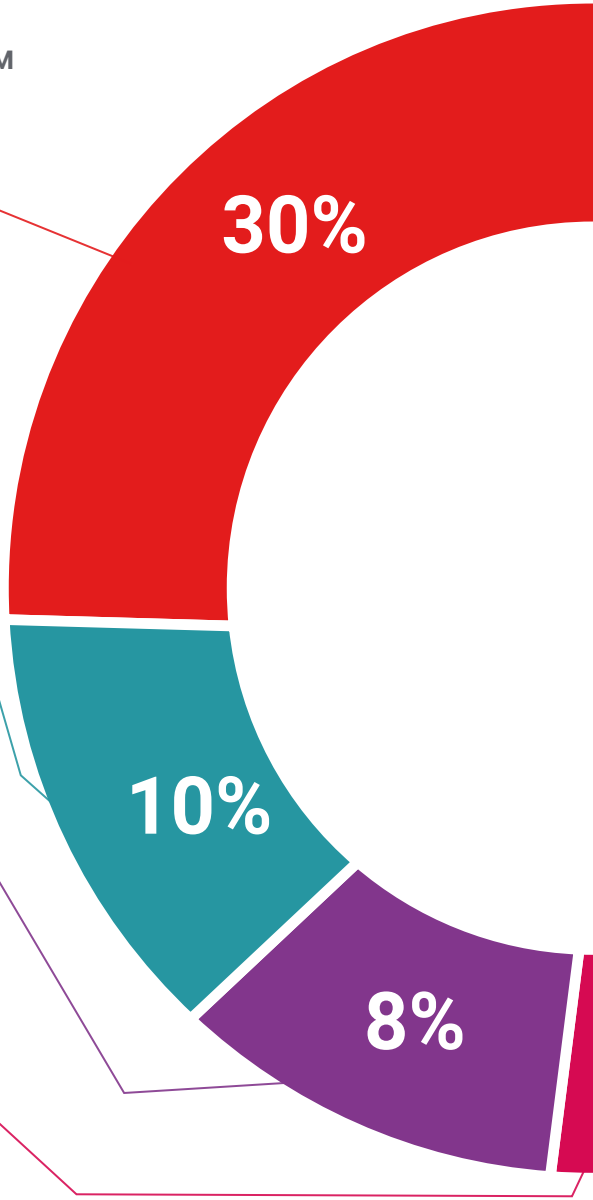
Так называемый метод обучения у эксперта укрепляет знания и память, а также формирует уверенность в наших будущих сложных решениях.

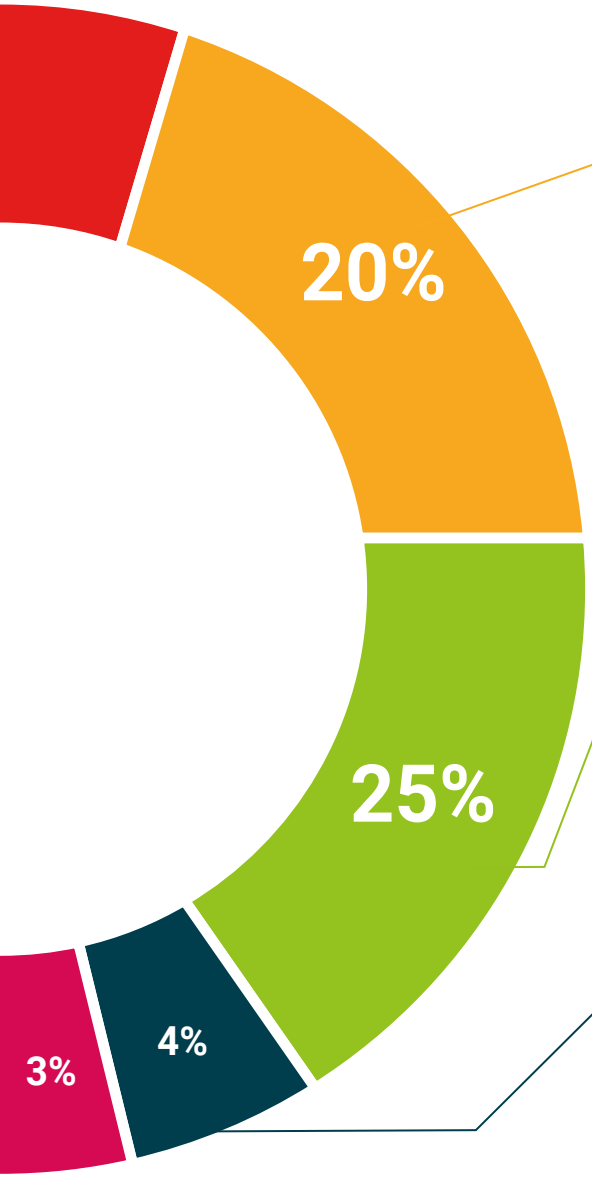
Практика навыков и компетенций

Студенты будут осуществлять деятельность по развитию конкретных компетенций и навыков в каждой предметной области. Практика и динамика приобретения и развития навыков и способностей, необходимых специалисту в рамках глобализации, в которой мы живем.

Дополнительная литература

Новейшие статьи, консенсусные документы и международные руководства включены в список литературы курса. В виртуальной библиотеке TECH студент будет иметь доступ ко всем материалам, необходимым для завершения обучения.





Метод кейсов

Метод дополнится подборкой лучших кейсов, выбранных специально для этой квалификации. Кейсы представляются, анализируются и преподаются лучшими специалистами на международной арене.

Интерактивные конспекты

Мы представляем содержание в привлекательной и динамичной мультимедийной форме, которая включает аудио, видео, изображения, диаграммы и концептуальные карты для закрепления знаний.

Эта уникальная обучающая система для представления мультимедийного содержания была отмечена компанией Microsoft как "Европейская история успеха".

Тестирование и повторное тестирование

На протяжении всей программы мы периодически оцениваем и переоцениваем ваши знания с помощью оценочных и самооценочных упражнений: так вы сможете убедиться, что достигаете поставленных целей.

06

Квалификация

Специализированная магистратура в области разработки программного обеспечения гарантирует, помимо самого строгого и современного обучения, получение диплома об окончании Специализированной магистратуры, выдаваемого TECH Global University.

Успешно пройдите эту программу и получите университетский диплом без хлопот, связанных с поездками и оформлением документов"

Данная программа позволит вам получить собственный диплом университета — **Специализированная магистратура в области разработки программного обеспечения**, одобренный **TECH Global University**, крупнейшим цифровым университетом в мире.

Tech Global University, является Официальным Европейским Университетом, признанным правительством Андорры (***официальный бюллетень***). Андорра является частью Европейского пространства высшего образования (ЕПВО) с 2003 года. ЕПВО — это инициатива, выдвинутая Европейским союзом с целью организации международной системы обучения и гармонизации систем высшего образования стран-участниц этого пространства. Проект способствует распространению общих ценностей, внедрению совместных инструментов и укреплению механизмов обеспечения качества для расширения сотрудничества и мобильности между студентами, исследователями и учеными.

Данный собственный диплом **Tech Global University** — европейская программа непрерывного обучения и повышения квалификации, которая гарантирует приобретение компетенций в своей области знаний, обеспечивая высокую учебную ценность для студента, прошедшего эту программу.

Диплом: **Специализированная магистратура в области разработки программного обеспечения**

Формат: **онлайн**

Продолжительность: **12 месяцев**

Аккредитация: **60 ECTS**


tech global university

Г-н/Г-жа _____ с документом, удостоверяющем личность, _____
успешно сдал(а) экзамен и получил(а) диплом:

Специализированная магистратура в области разработки программного обеспечения

Данная собственная программа университета рассчитана на 1.800 часов, что эквивалентно 60 ECTS, с датой начала обучения дд/мм/гггг и датой окончания дд/мм/гггг.

TECH Global University — университет, официально признанный правительством Андорры 31 января 2024 года и входящий в Европейское пространство высшего образования (ЕПВО).

В Андорре-ла-Велья, 28 февраля 2024 года

Д-р Pedro Navarro Illana
Ректор

unique TECH code: AFWOR23S techtitute.com/certificates


Специализированная магистратура в области разработки программного обеспечения

Общее распределение учебного плана

Тип предмета	Кредиты ECTS
Обязательный (Обязат.)	60
Необязательный (Необязат.)	0
Внешние стажировки (ВС)	0
Магистерская диссертация (МД)	0
	Всего 60

Общее распределение учебного плана

Курс	Предмет	ECTS	Характер
1º	Основы программирования	6	Обязат
1º	Структура данных	6	Обязат
1º	Алгоритм и сложность	6	Обязат
1º	Базы данных	6	Обязат
1º	Расширенные базы данных	6	Обязат
1º	Продвинутое проектирование алгоритмов	6	Обязат
1º	Взаимодействие человека и компьютера	6	Обязат
1º	Расширенное программирование	6	Обязат
1º	Разработка сетевых приложений	6	Обязат
1º	Программная инженерия	6	Обязат

Д-р Pedro Navarro Illana
Ректор

tech global university

*Гаагский апостиль. В случае, если студент потребует, чтобы на его диплом в бумажном формате был проставлен Гаагский апостиль, TECH Global University предпримет необходимые шаги для его получения за дополнительную плату.

tech global university

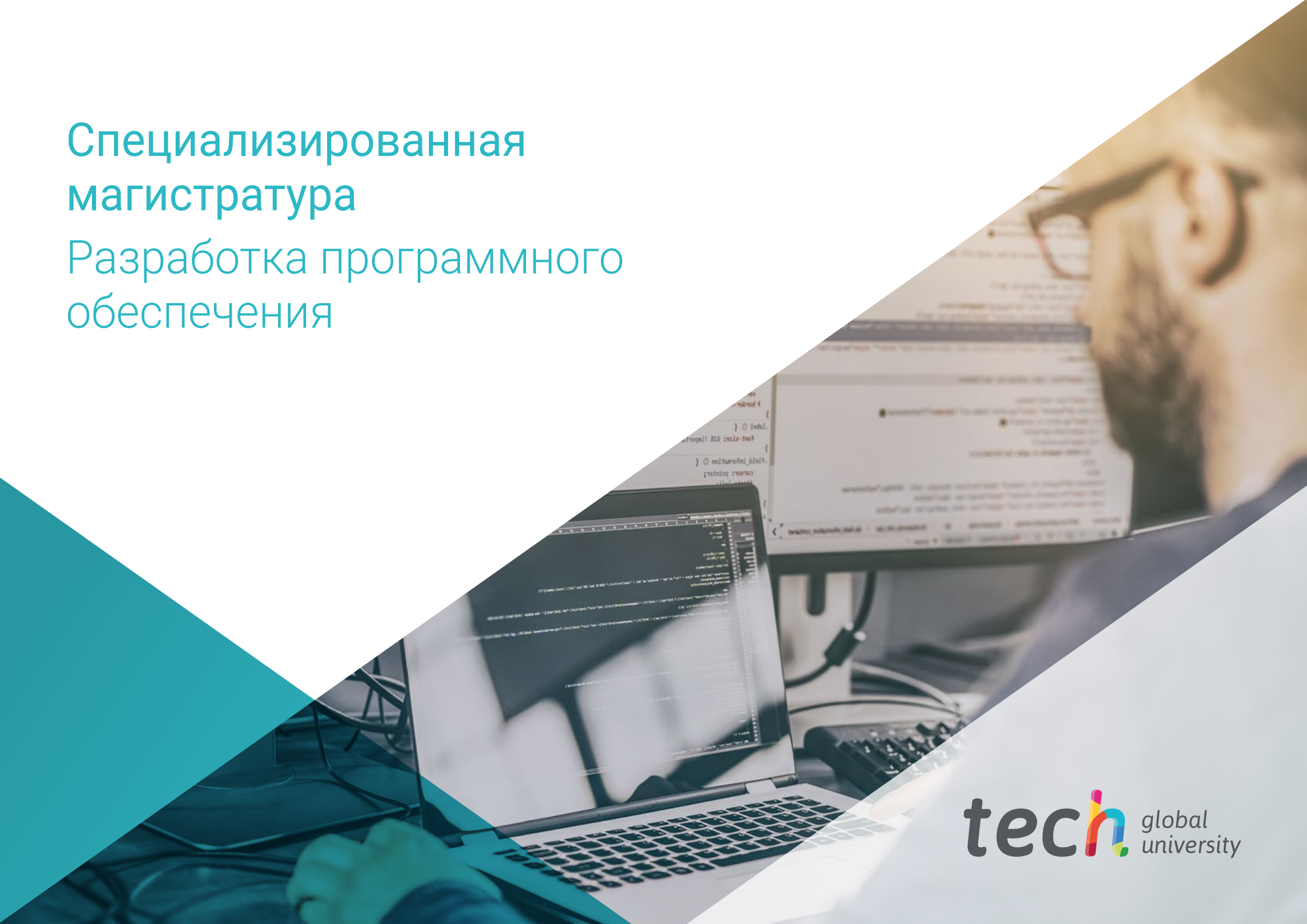
Специализированная магистратура

Разработка программного обеспечения

- Формат: **онлайн**
- Продолжительность: **12 месяцев**
- Учебное заведение: **TECH Global University**
- Аккредитация: **60 ECTS**
- Расписание: **по своему усмотрению**
- Экзамены: **онлайн**

Специализированная магистратура

Разработка программного обеспечения



tech global university