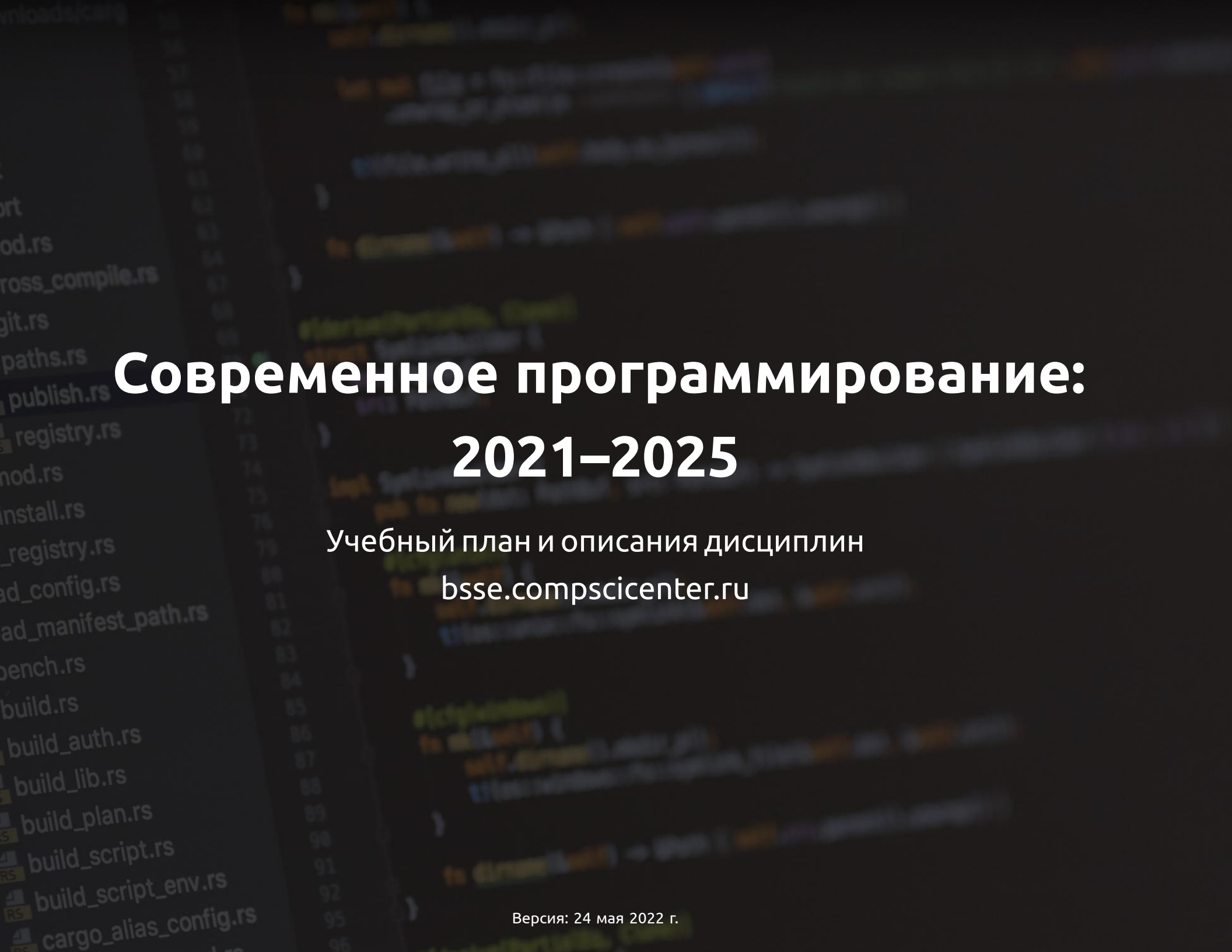


Современное программирование: 2021–2025

Учебный план и описания дисциплин

bsse.compscicenter.ru

Версия: 24 мая 2022 г.

О программе

Мы работаем на факультете математики и компьютерных наук (МКН) СПбГУ при поддержке компании JetBrains и готовим программистов высокой квалификации, способных решать сложные задачи в любых областях информационных технологий.

Вот как мы это делаем. Во-первых, мы считаем, что будущим программистам в первую очередь нужны:

- крепкая математика и теоретическая информатика;
- умение писать качественные программы;
- знание алгоритмов.

Это база, мы формируем её первые два года, сотрудничая с лабораторией Чебышёва и ПОМИ РАН, ведущими отечественными научными центрами.

Во-вторых, студентам никак не обойтись без:

- изучения современных технологий;
- участия в проектах по разработке ПО.

Мы хорошо учим современным технологиям, потому что эти курсы у нас читают разработчики ведущих IT-компаний и опытные преподаватели Computer Science центра. Студенты участвуют в реальных проектах под руководством профессионалов отрасли. Такие проекты дают опыт работы в условиях, максимально похожих на «боевые». Заодно можно изучить массу новых технологий. А ещё проекты могут быть исследовательскими. В программировании много чего можно исследовать!

В-третьих, начиная с третьего курса у студентов появляется большой выбор дисциплин: можно углубляться в программирование в интересных областях, а можно и переключиться в математику (математические спецкурсы и научные семинары в том же здании!).

В-четвёртых, у нас маленькие наборы (30 человек и две группы на практических занятиях) — это идеальные условия для передачи знаний, ведь преподаватель может посмотреть код каждого студента и поругать его на code review!

Что ещё? Можно участвовать в соревнованиях (олимпиадная математика, спортивное программирование, турниры по машобучу) — у нас отличные тренеры! Наконец, мы учимся на Ваське (это исторический центр Петербурга, Васильевский остров).

В общем, у нас хорошо.

Совет

Информационные технологии развиваются стремительно. Совет образовательной программы одобряет все изменения в ней и следит за тем, чтобы давались актуальные знания, востребованные в науке и индустрии. В Совет входят и действующие учёные, и профессиональные разработчики программного обеспечения — как из России, так и из других стран.

Андрей Андреевич Бреслав

Руководитель проекта Kotlin, JetBrains.

Андрей Владимирович Иванов

Старший вице-президент по инвестициям, исследованиям и образованию в JetBrains. Председатель Совета.

Владислав Анатольевич Казначеев

Руководитель группы разработки пользовательского интерфейса операционной системы Chrome OS, Google.

Дмитрий Игоревич Качмар

Коммерческий директор, Яндекс.

Кирилл Владимирович Кринкин

К.т.н., доцент, зав. каф. математического обеспечения и применения ЭВМ Санкт- Петербургского электротехнического университета «ЛЭТИ».

Александр Сергеевич Куликов

Д.ф.-м.н., с.н.с. ПОМИ РАН, профессор и руководитель ОП «Современное программирование» в СПбГУ, член совета Computer Science центра, координатор образовательных проектов в JetBrains.

Михаил Владимирович Левин

Директор по машинному интеллекту ООО «Яндекс. Маркет».

Илья Лазаревич Миронов

Ведущий научный сотрудник, Facebook.

Павел Аркадьевич Певзнер

Зав. лаб. «Центр алгоритмической биотехнологии» Института трансляционной биомедицины СПбГУ, профессор Калифорнийского университета (Сан-Диего, США).

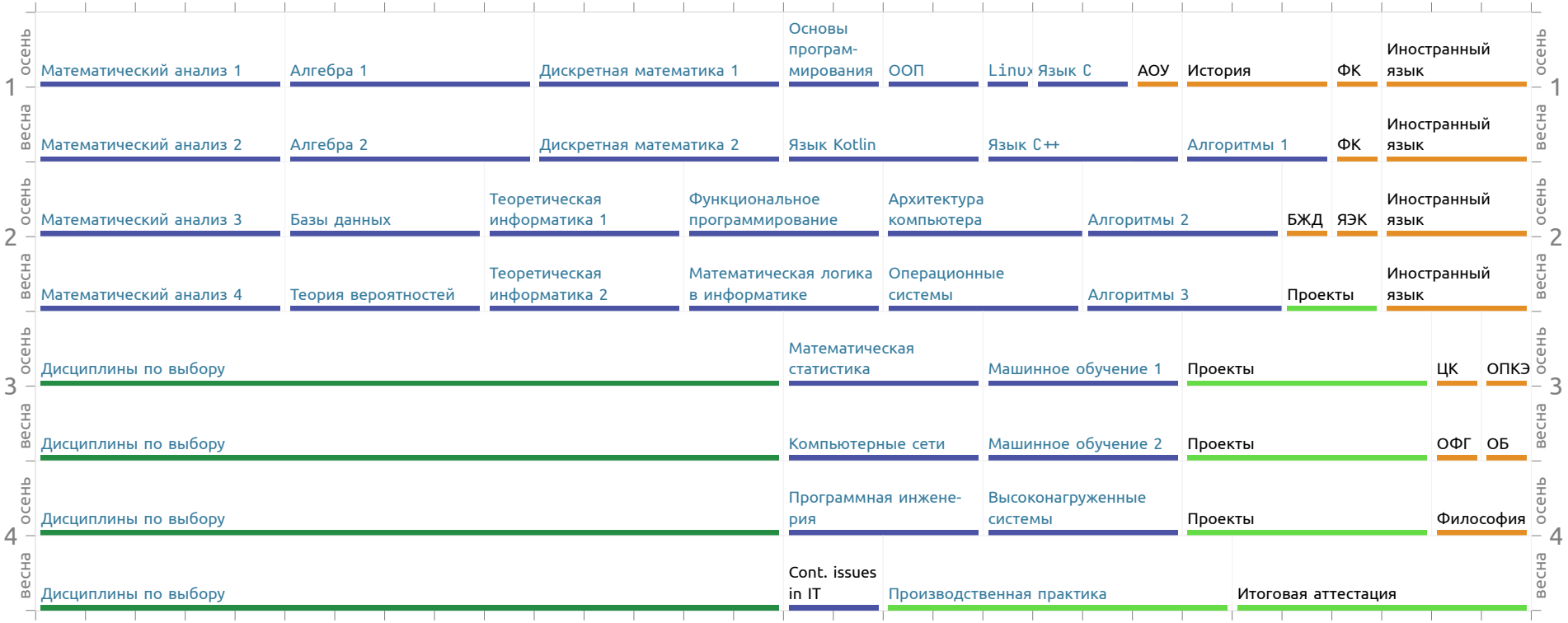
Станислав Константинович Смирнов

Лауреат Филдсовской премии, руководитель лаборатории имени П. Л. Чебышёва СПбГУ, профессор Женевского университета (Швейцария).

Андрей Николаевич Терехов

Д.ф.-м.н., профессор, зав. каф. системного программирования СПбГУ.

Учебный план



Учебный план определяет практически всё, чем занимаются студенты на протяжении всего времени обучения — в бакалавриате это четыре года (восемь семестров) — изучают обязательные дисциплины и дисциплины по выбору, участвуют в проектах, проходят производственную практику и готовят выпускную работу. Схема учебного плана отражает объёмы этих видов деятельности в зачётных единицах: чем шире прямоугольник на схеме, тем больше времени отводится на соответствующую деятельность. Для учебных дисциплин это время на аудиторные занятия, самостоятельную работу в семестре и подготовку к экзамену, если он предусмотрен. К примеру, у дисциплины на четыре зачётные единицы обычно две пары занятий в неделю в течение семестра. Четверть всего обучения — это дисциплины по выбору студента. В каждом из семестров с пятого по восьмой можно выбрать четыре–пять таких дисциплин в зависимости от их объёма. Конкретный перечень предлагаемых дисциплин по выбору может каждый год меняться.

По ссылкам в названиях дисциплин можно перейти к описанию конкретной дисциплины.

Содержание

Математический анализ 1

I курс, сентябрь–декабрь.
5 зачётных единиц, 3 часа лекций, 3 часа практик в неделю.
зачёт, экзамен.

Аннотация

Первая часть курса математического анализа состоит в изучении базовых понятий и теорем курса дифференциального исчисления функций одной вещественной переменной. Однако помимо изучения аксиоматики вещественных чисел, свойств функций и стандартных теорем начального курса математического анализа, слушателям предлагается изучить основные понятия (предел и непрерывность) в более абстрактных ситуациях метрического и топологического пространства, что позволяет сделать первую часть курса более глубокой и насыщенной. Тем не менее, дифференциальное исчисление в первом семестре излагается лишь для функций одной вещественной переменной: его основной целью является обучение крайне важным для математической культуры умениям — отыскание экстремумов функций с помощью принципа Ферма и приближенное вычисление значений с помощью формулы Тейлора. Эта часть курса иллюстрируется рядом важных и интересных приложений (вычисление асимптотик сумм, доказательство иррациональности чисел π и e, доказательство классических неравенств). Завершает первую часть курса вводное построение интеграла.

Пререквизиты

- Дискретная математика 1.
- Алгебра 1.

Где понадобится

- Во всех последующих математических дисциплинах аналитического характера.
- Для понимания курсов: Математический анализ 2, 3 и 4, Теория вероятностей, Математическая статистика, Выпуклая оптимизация, Машинное обучение 1.
- Для формирования общей математической культуры.

Содержание

- Аксиоматика вещественных чисел.
- Метрические, топологические и нормированные пространства. Пределы и непрерывность, компактность. Теоремы Вейерштрасса и Кантора.
- Дифференциальное исчисление функций одной вещественной переменной (теоремы Ролля, Ферма, Коши и Лагранжа, формула Тейлора).
- Интегрирование функций одной вещественной переменной (первообразная и определённый интеграл, существование и свойства).

Вы научитесь

- оценивать качественно и количественно поведение функций и сумм;
- основным принципам исследования функций;
- оперировать абстрактными структурами и оценивать возможность обобщения утверждений на более широкий класс пространств.

Алгебра 1

I курс, сентябрь–декабрь.
5 зачётных единиц, 2 часа лекций, 2 часа практик в неделю.
зачёт, экзамен.

Аннотация

Программа включает традиционный материал начального курса высшей алгебры: общее знакомство с основными алгебраическими структурами, комплексные числа, свойства кольца многочленов одной переменной, важнейшие понятия линейной алгебры как на координатном, так и на абстрактном уровне.

Курс содержит большое количество практических задач, призванных дать возможность слушателям закрепить полученные знания различных алгебраических методов, которые будут возникать практически во всех дальнейших курсах — как математических, так и прикладных.

Пререквизиты

- Математика на уровне школьной программы.

Где понадобится

- Во всех последующих математических дисциплинах.
- Для понимания курсов: Алгебра 2, Математический анализ 1, 2, Алгоритмы и структуры данных 1, Машинное обучение 1.

Содержание

- Алгебраические структуры: группы, кольца, поля.
- Комплексные числа: тригонометрическая форма, формула Муавра, извлечение корней.
- Многочлены и рациональные дроби: евклидовы кольца, разложение на неприводимые, кратные корни, разложение дроби в сумму простейших, интерполяция.
- Векторные пространства и линейные системы: линейная зависимость, ранг матрицы, метод Гаусса, теорема Кронекера–Капелли, линейные отображения, размерность ядра и образа, суммы и пересечения подпространств.
- Определители: свойства, минорный ранг, взаимная матрица, теорема Крамера.

Вы научитесь

- оперировать с комплексными числами, многочленами, рациональными дробями, линейными пространствами, матрицами;
- раскладывать многочлены на неприводимые множители, находить кратные корни, раскладывать рациональные дроби в сумму простейших дробей;
- вычислять ранги матриц, определители, анализировать системы линейных уравнений и решать их методом Гаусса, находить суммы и пересечения линейных подпространств.

Дискретная математика 1

I курс, сентябрь–декабрь.
5 зачётных единиц, 2 часа лекций, 2 часа практик в неделю.
экзамен.

Аннотация

Одна из основных целей данного курса — научиться строить и строго анализировать математические модели, возникающие в программировании и компьютерных науках. Много внимания будет уделено формальным доказательствам корректности: разберёмся, какие бывают методы доказательства; узнаем, как доказывать существование, оптимальность и универсальность; увидим много примеров того, как интуитивно кажущееся правильным доказательство оказывается формально неверным (что на практике может привести к сбою оборудования или программного обеспечения). Также узнаем, как эффективно перебирать различные комбинаторные объекты и вычислять их количество, не перечисляя их явно. Увидим, как часто это помогает при построении эффективных алгоритмов. Наконец, начнём изучать теорию вероятностей — раздел математики, возникающий в самых разных областях компьютерных наук: в алгоритмах, теории игр, криптографии, обработке сигналов и других. Узнаем, как формально определять и оценивать вероятности различных случайных процессов. Научимся пользоваться стандартными методами оценки разных параметров случайных величин — среднего, дисперсии, уклонений.

Курс содержит большое количество задач разной степени сложности — как на доказательство, так и на программирование. Задачи на доказательство призваны дать слушателям возможность потренироваться кратко и строго записывать рассуждения. Задачи на программирование покажут, как именно возникающие дискретные структуры используются на практике, и помогут разобраться во всех деталях этих структур.

Пререквизиты

- Математика: школьная программа (доказательства, основные функции: логарифм, многочлен, экспонента).
- Программирование: базовое владение языком программирования `Python` (ввод-вывод, циклы, рекурсия).

Где понадобится

- Во всех последующих математических дисциплинах.
- Для построения и анализа алгоритмов и эффективных программ.
- Для понимания курсов: Алгоритмы и структуры данных, Теория вероятностей, Теоретическая информатика, Математическая статистика, Машинное обучение.

Содержание

- Доказательства: существование, оптимальность, универсальность.
- Множества: мощности, диагональный аргумент, порядки, цепи/антицепи.
- Перебор и подсчёт: подмножества, перестановки, размещения, сочетания, скобочные последовательности, разбиения, рекуррентные соотношения.
- Вероятность: события, условная вероятность и независимость, случайные величины, уклонения, вероятностный метод.

Вы научитесь

- строго и коротко записывать доказательства;
- оценивать время работы алгоритмов и программ;
- оценивать вероятности;
- применять идеи дискретной математики в разных областях IT;
- реализовывать перебор, кодирование и шифрование, сведения к задаче выполнимости.

Основы программирования

I курс, сентябрь-октябрь.
2 зачётные единицы, 2 часа лекций, 2 часа практик в неделю.
зачёт.

Аннотация

Первокурсники обычно уже знакомы с программированием. Многие участвовали в олимпиадах, кто-то имел опыт реализации прикладных проектов. Практика, впрочем, показывает, что при обучении программированию на предыдущих этапах мало внимания уделяется таким вопросам как профессиональное отношение программиста к работе, качество кода, суть языка программирования как промышленного инструмента со своими достоинствами и недостатками.

Этот курс разработан с целью осветить все эти вопросы, а также подготовить студентов к работе на последующих программистских курсах путём формирования у них некоторой общей базы знаний в области разработки ПО. В качестве основного языка программирования на курсе будет использоваться язык `Kotlin`. По ходу курса студенты реализуют несколько программных проектов и благодаря многократным обзорам кода (code review) получат навыки написания кода, решающего поставленные задачи и удовлетворяющего при этом критериям качества.

Пререквизиты

- Компиляция и запуск программы на любом языке программирования.
- Работа с массивами.
- Обработка строк.
- Представление о структурах данных.
- Использование подпрограмм.

Где понадобится

- Во всех последующих дисциплинах по программированию.
- В командных проектах.
- В работе на должности разработчика программного обеспечения.

Содержание

- Программирование как профессия: основные составляющие, мастерство в профессии и качество кода.
- Язык `Kotlin`: базовые типы данных, переменные, выражения и операторы, функции, собственные типы данных, ввод-вывод, контейнерная библиотека, исключения.
- Технологии: контроль версий и удалённое хранение кода, именование, декомпозиция, тестирование и отладка, рефакторинг.

Вы научитесь

- писать поддерживаемый код промышленного качества;
- использовать профессиональный инструментарий;
- хранить код под управлением систем контроля версий;
- тестировать код;
- применять механизмы обработки исключений для написания устойчивых к ошибкам ввода программ.

Объектно-ориентированное программирование

I курс, ноябрь–декабрь.
2 зачётные единицы, 2 часа лекций, 2 часа практик в неделю.
зачёт.

Аннотация

Почему в 1968 году оператор перехода `goto` был объявлен вредным? Почему корректно работающий код может, тем не менее, быть единодушно признан ужасным?

Подобно тому как правила литературного языка не являются гарантией ясности и хорошего стиля, одни только правила языка программирования не обеспечивают создания качественного кода. Языки дополняются приёмами их правильного использования. Иногда из этих приёмов возникают методологии, которые становятся основой более современных языков. Самой известной из таких методологий является объектно-ориентированное программирование.

Студенты изучат такие принципы ООП, как абстракция, инкапсуляция, наследование и полиморфизм, изучат основные шаблоны проектирования.

В качестве языка программирования на курсе будет использоваться язык `Kotlin`. Практическая часть включает реализацию нескольких небольших программных проектов, предполагающих проектирование объектных моделей, создание предметно-зависимых языков (DSL) и преобразование кода при изменении требований.

Пререквизиты

- Представление о структурах данных.
- Основы программирования.

Где понадобится

- Во всех последующих дисциплинах по программированию.
- В командных проектах.
- В работе на должности разработчика программного обеспечения.

Содержание

- ООП: абстракция, инкапсуляция, наследование, полиморфизм.
- `Kotlin`: классы, интерфейсы, обобщения, функции-расширения, лямбда-выражения.
- Принципы проектирования: единственность ответственности, инверсия зависимостей, разделение интерфейсов.

Вы научитесь

- строить объектные модели для разных предметных областей;
- писать код используя шаблоны проектирования;
- создавать предметно-зависимые языки.

Основы Linux

I курс, сентябрь-октябрь.
1 зачётная единица, 2 часа практик в неделю.
зачёт.

Аннотация

Этот практический курс посвящён освоению командной строки как основного инструмента при использовании компьютеров. Множество действий можно выполнять быстрее и эффективнее, если описать их как последовательность вызовов программ и оформить в виде сценария на языке командного интерпретатора. Операционная система Linux идеально подходит для изучения такого стиля работы, поскольку содержит в себе все необходимые инструменты. Как выяснится по ходу курса, программистам часто приходится обращаться к командной строке. После этого курса студенты смогут делать это легко и с удовольствием.

Пререквизиты

- Базовое владение любым языком программирования.

Где понадобится

- В программистских дисциплинах, ориентирующихся на использование командной строки.
- В курсе операционных систем.
- Для администрирования своего компьютера.
- В рамках проектов, требующих особых условий для сборки и доставки пользователю.

Содержание

- Работа в командной строке: доступ к файловой системе, управление процессами, запуск программ и их взаимодействие, переменные окружения, важнейшие утилиты командной строки.
- Программирование на языке bash: переменные, ввод-вывод, управляющие инструкции, функции.
- Элементы DevOps: сборка ПО, контейнеризация, интеграция и доставка.

Вы научитесь

- решать задачи путём комбинирования имеющихся программ;
- применять консольные инструменты разработки программ;
- обрабатывать текстовые данные;
- автоматизировать действия по обслуживанию компьютерных систем.

Программирование на языке C

I курс, ноябрь–декабрь.
2 зачётные единицы, 2 часа лекций, 2 часа практик в неделю.
экзамен.

Аннотация

Конструкции языка C близко сопоставляются типичным машинным инструкциям, благодаря чему он традиционно является одним из основных языков системного программирования (в частности, на нем написана ОС Linux). Изучение приемов программирования на C позволяет не только научиться писать эффективные программы, но и разобраться, как работает компьютер при их выполнении.

Язык C повлиял на синтаксис таких языков, как C++, C#, Java и Objective-C. Знание C позволит лучше понять решения, принятые авторами этих языков, и оценить накладные расходы по времени и памяти при их использовании.

Пререквизиты

- Базовое владение любым языком программирования (переменные, ветвления, циклы, функции, массивы).
- Основы Linux: навыки пользователя.

Где понадобится

- Для понимания курсов: язык программирования C++, операционные системы.
- При программировании встраиваемых систем (embedded systems).

Содержание

- Процесс превращения программы в машинный код (компилятор, ассемблер, линковщик).
- Указатели.
- Как устроен вызов функции?
- Карта памяти: стек и куча.
- Указатели на функции. Интрузивные списки.
- Обзор стандартной библиотеки libc.

Вы научитесь

- читать и писать полезные программы на языке C;
- детально понимать процесс построения программы из исходных кодов (gcc, make);
- детально понимать процесс выполнения программы;
- использовать библиотеки.

Математический анализ 2

I курс, февраль–май.
5 зачётных единиц, 3 часа лекций, 3 часа практик в неделю.
зачёт, экзамен.

Аннотация

Вторая часть курса математического анализа начинается с приложений интегрального исчисления функций одной вещественной переменной. В частности, изучаются методы нахождения площадей, объемов и длин, анализ сумм с помощью приближенных методов численного интегрирования, приближенное вычисление сумм и произведений. Отдельное внимание уделено несобственным интегралам. Основную часть курса составляет дифференциальное исчисление функций нескольких вещественных переменных. Главной целью является исследование функций нескольких переменных на экстремум, решение оптимизационных задач, в частности, метод градиентного спуска. Большая часть материала даётся в абстрактной ситуации нормированного пространства, подразумевающей содержательные и интересные приложения в задачах, постановка которых предполагает наличие бесконечного количества степеней свободы (задача о брахистохроне, задача о цепной линии, изопериметрическая задача), в частности, курс содержит элементы вариационного исчисления и дифференциальных уравнений. Такой подход предполагает более подробное изучение конкретных нормированных пространств, состоящих из непрерывных и дифференцируемых функций, что дает правильное и глубокое понимание равномерной сходимости и ее важности в теоремах о перестановочности основных аналитических операций. Основные трудные теоремы курса выводятся из принципа сжимающих отображений — одного из самых мощных и идейных инструментов доказательства теорем существования. Завершает курс второго семестра теория числовых и функциональных рядов, предваряющих изучение теории меры в третьем семестре.

Пререквизиты

- Математический анализ 1.
- Дискретная математика 1.
- Алгебра 1 и 2.

Где понадобится

- Во всех последующих математических дисциплинах аналитического характера.
- Для понимания курсов: Математический анализ 3 и 4, Теория вероятностей, Математическая статистика, Выпуклая оптимизация, Машинное обучение 1.
- Для создания общей математической культуры.

Содержание

- Приложения определённого интеграла: площадь, объем, длина, численное интегрирование, формула трапеций, формула Стирлинга.
- Несобственные интегралы.
- Анализ в нормированных пространствах: принцип сжимающих отображений, дифференцируемость, теорема о неявной функции.
- Задачи оптимизации: расстояние до подпространства, минимум квадратичной формы на сфере, изопериметрическая задача, задача о брахистохроне, задача о цепной линии.
- Теория числовых и функциональных рядов.

Вы научитесь

- применять интегральное исчисление на практике;
- оценивать качественно и количественно поведение функций нескольких переменных;
- решать простейшие задачи оптимального управления.

Алгебра 2

I курс, февраль–май.
5 зачётных единиц, 2 часа лекций, 2 часа практик в неделю.
зачёт, экзамен.

Аннотация

Вторая часть курса алгебры носит более теоретический и менее вычислительный характер, включает ряд красивых теорем. Знакомство с теорией групп создаёт основу для изучения других разделов высшей алгебры и теории чисел, способствует развитию абстрактного мышления. Дополнительные сведения из линейной алгебры используются, в частности, в классических алгоритмах на графах и в алгоритмах распознания образов. Курс выключает также основы теории конечных полей, применяемой в криптографии.

Одна из главных целей курса — развить у слушателей абстрактное мышление, которое позволит и в дальнейшем замечать алгебраические структуры в самых разных областях — базах данных, компиляторах и языках программирования, теории графов. Вторая цель — дать необходимую практику применения методов алгебры, которые будут использоваться в дальнейших прикладных курсах — алгоритмах, теории вероятностей, машинном обучении и других.

Пререквизиты

- Алгебра 1.
- Дискретная математика 1.

Где понадобится

- Практически во всех последующих математических дисциплинах: математический анализ 2, 3 и 4, алгоритмы и структуры данных 1, машинное обучение 1.

Содержание

- Теория групп: подгруппы, гомоморфизмы, факторгруппы, симметрические группы, свободная группа.
- Линейные операторы: собственные значения, характеристический многочлен, канонические формы.
- Квадратичные формы: ортогонализация, закон инерции.
- Аналитическая геометрия: евклидовы пространства, ортогональное дополнение, каноническая форма нормального оператора.
- Конечные поля: классификация, цикличность мультипликативной группы.

Вы научитесь

- выполнять вычисления в группе перестановок, задавать группы образующими и соотношениями, вычислять факторгруппы;
- уметь находить собственные значения линейных операторов, их алгебраические и геометрические кратности;
- уметь находить каноническую форму ортогонального оператора, вычислять расстояния между линейными многообразиями.

Дискретная математика 2

I курс, февраль–май.
5 зачётных единиц, 2 часа лекций, 2 часа практик в неделю.
экзамен.

Аннотация

Во второй части курса по дискретной математике мы изучим графы, коды и модели вычислений. Узнаем, как самые разные практические задачи формулируются на языке графов и как пользоваться стандартными методами области теории графов для решения таких задач. Увидим, какие бывают требования к методам кодирования, изучим стандартные конструкции и их ограничения. Наконец, затронем различные модели вычислений и их ограничения.

Для закрепления материала будем решать много задач — как теоретических, так и программистских.

Пререквизиты

- Дискретная математика 1: доказательства, индукция, комбинаторика.
- Программирование: базовое владение языком программирования `Python`.

Где понадобится

- Для представления и анализа различных данных в виде графов.
- В теории формальных языков, при реализации компиляторов.
- Для понимания того, какие задачи могут быть эффективно решены на практике, и того, какие есть ограничения для решения трудных задач.

Содержание

- Графы: деревья; эйлеровы и гамильтоновы циклы; связность, разрезы и потоки; паросочетания и покрытия; раскраски; планарность.
- Игры и стратегии.
- Коды с минимальной избыточностью и коды, исправляющие ошибки.
- Модели вычислений и классы сложности.

Вы научитесь

- формулировать возникающие на практике задачи в терминах теории графов;
- пользоваться библиотечными методами для вычисления различных параметров графов;
- различать вычислительно простые и трудные задачи;
- оценивать вычислительные ресурсы, необходимые для решения задач;
- знать ограничения различных вычислительных моделей.

Алгоритмы и структуры данных 1

I курс, февраль–май.
3 зачётных единицы, 2 часа лекций, 2 часа практик в неделю.
зачёт.

Аннотация

Основная цель данного курса — познакомить студентов со стандартными методами решения наиболее часто возникающих на практике задач. Студенты научатся формально оценивать количество используемых алгоритмом ресурсов (время и память); изучат разные способы хранения и обработки информации (структуры данных), и научатся подбирать оптимальную структуру данных в зависимости от требуемой функциональности; увидят примеры применения жадного метода и метода динамического программирования, а также потренируются решать задачи этими методами.

Конечно, для большинства включенных в курс алгоритмов и структур данных несложно найти уже готовую реализацию. Тем не менее, знание принципов работы этих алгоритмов, их преимуществ и недостатков, области применения и ограничений, позволит студенту в будущем быстро и эффективно подбирать готовое (или реализовывать своё) решение к каждой конкретной задаче. В том числе поэтому важная часть курса — это строгие доказательства корректности работы изучаемых алгоритмов, а также получение строгих оценок на время их работы. На практических занятиях студенты будут упражняться в применении изученных на лекциях алгоритмов в различных ситуациях; иногда при этом потребуется модифицировать изученный алгоритм, или воспользоваться идеями из доказательства корректности его работы для получения нового алгоритма. Кроме того, студенты попробуют реализовать изученные алгоритмы на практике и решить с их помощью ряд модельных задач.

Пререквизиты

- Дискретная математика 1: дискретная теория вероятностей.
- дискретная математика 2: основы теории графов.
- Алгебра 1: линейная алгебра, матрицы.
- Программирование на языке C++: базовое владение языком программирования C/C++.

Где понадобится

- Для написания эффективного и надёжного кода.
- Для прохождения технических интервью.
- В работе на должности разработчика программного обеспечения.

Содержание

- Сложность алгоритмов: O-символика, рекуррентные соотношения.
- Базовые алгоритмы и структуры данных: список, динамический массив, двоичный поиск, сортировки, двоичная куча, хеш-таблица.
- Алгоритмы на графах: поиск в глубину и его приложения, алгоритмы поиска кратчайших путей.
- Жадные алгоритмы: алгоритм Хаффмана, алгоритм Белади, алгоритмы Прима и Краскала.
- Динамическое программирование.

Вы научитесь

- оценивать время работы алгоритмов;
- сравнивать эффективность разных решений одной и той же задачи;
- строго доказывать корректность алгоритмов;
- реализовывать и применять изученные алгоритмы и структуры данных.

Язык программирования C++

I курс, февраль-май.
4 зачётные единицы, 2 часа лекций, 2 часа практик в неделю.
экзамен.

Аннотация

Язык C++ является де-факто стандартом для написания высокопроизводительных программ. С одной стороны, он позволяет эффективно использовать ресурсы компьютера, а с другой — позволяет писать программы с использованием наиболее популярных подходов современного программирования (процедурного, объектно-ориентированного, обобщенного, метапрограммирования и других).

Целью курса является обучение написанию на C++ больших программ и библиотек, которые можно переиспользовать в других проектах. Особенностью курса является то, что очередной механизм языка рассматривается только после того, как слушатели смогут написать его сами.

Пререквизиты

- Программирование на языке C.
- Базовое знание ООП на любом языке.

Где понадобится

- В работе разработчика ПО на C++.
- В соревнованиях по спортивному программированию и курсах по алгоритмам.

Содержание

- Разработка объектно-ориентированных программ на C++.
- Обработка ошибок (в том числе, исключения).
- Разработка библиотек с помощью обобщенного программирования (шаблоны) и метапрограммирования.
- Обзор стандартной библиотеки STL.
- Элементы многопоточного программирования.
- Элементы проектирования.
- Обзор последних версий стандарта C++.

Вы научитесь

- читать и писать программы на современном C++;
- проектировать и реализовывать код, который можно использовать повторно;
- использовать все возможности библиотеки STL.

Программирование на языке Kotlin

I курс, февраль–май.
4 зачётные единицы, 1 час лекций, 2 часа практик в неделю.
зачёт.

Аннотация

Kotlin — это статически типизированный язык программирования, разрабатываемый компанией JetBrains с 2010 года. Назван в честь острова Котлин, на котором расположен город Кронштадт. Авторы ставили целью создать язык более лаконичный и типобезопасный, чем Java, и более простой, чем Scala. Следствием упрощения по сравнению со Scala стали также более быстрая компиляция и лучшая поддержка языка в IDE.

Наибольшую популярность язык приобрёл при разработке приложений под Android и серверных приложений. Также возможна компиляция в JavaScript и native код для многих платформ.

Kotlin позволяет писать в различных парадигмах: объектно-ориентированной, функциональной, императивной и пр. Синтаксис языка позволяет создавать собственные предметно-зависимые языки - domain specific languages (DSL). Отдельного упоминания заслуживает поддержка асинхронного программирования на Kotlin с помощью корутин.

Студенты изучат все тонкости синтаксиса языка Kotlin, и, реализовав несколько программных проектов, освоят все связанные с языком инструменты разработки.

Пререквизиты

- Представление о структурах данных.
- Введение в программирование.

Где понадобится

- Во всех последующих дисциплинах по программированию.
- В командных проектах.
- В работе на должности разработчика программного обеспечения.

Содержание

- `Kotlin`: классы, интерфейсы, обобщения, функции-расширения, лямбда-выражения, reflection, и пр.
- Инструменты: сборка, отладка, профилирование и пр.
- Платформы: JVM, Android, native, JS, multiplatform

Вы научитесь

- писать серверные приложения;
- писать многопоточный и асинхронный код;
- создавать предметно-зависимые языки;

Математический анализ 3

II курс, сентябрь–декабрь.
5 зачётных единиц, 3 часа лекций, 2 часа практик в неделю.
зачёт, экзамен.

Аннотация

В третьей части курса математического анализа студенты изучают теорию меры и интеграл Лебега. В чистом виде, теория меры — это математический язык, очень глубоко формализующий привычное нам понятие объёма. Этот раздел математики лежит в основе нескольких других важных разделов — теория вероятностей, анализ Фурье и другие. Цель первой части данного курса — познакомиться с основными понятиями и теоремами теории меры и научиться в них свободно ориентироваться.

Цель второй части курса — познакомиться с некоторым конкретным примерам интегрирования. Заводится и исследуется понятие дифференциальной формы, исследуются асимптотики некоторых интегральных выражений с помощью B и Γ-функции. Эти математические конструкции будут в дальнейшем использоваться, например, для описания некоторых классических вероятностных распределений.

Курс сопровождается большим количеством практических задач, призванных помочь слушателям развить интуицию и закрепить изученный материал.

Пререквизиты

- Математический анализ 1 и 2.
- Алгебра 1.
- Дискретная математика 1 и 2: комбинаторика, мощности множеств, диагональный аргумент.

Где понадобится

- Для понимания курсов: Математический анализ 4, Теория вероятностей, Математическая статистика.

Содержание

- Теория меры: структуры множеств, объем и мера, произведение мер, стандартное продолжение мер, мера Лебега, измеримые функции, различные виды сходимости.
- Интеграл Лебега: построение интеграла Лебега и его свойства, перестановка предела и интеграла, принцип Кавальери, связь кратного и повторного интегралов, замена переменной в кратном интеграле.
- Интегралы с параметром и криволинейные интегралы: равномерная сходимость, дифференцирование по параметру, B и Γ-функции, криволинейные интегралы, дифференциальные формы первого порядка, замкнутые и точные формы, интегралы по гомотопным путям, формула Грина.

Вы научитесь

- хорошо ориентироваться в основных конструкциях теории меры;
- использовать свойства интеграла Лебега для вычисления интегральных выражений;
- писать стандартные интегральные оценки;
- базовым приёмам нахождения асимптотик интегральных выражений.

Теоретическая информатика 1: формальные языки

II курс, сентябрь–октябрь.
2 зачётные единицы, 2 часа лекций, 2 часа практики в неделю.
зачёт, экзамен.

Аннотация

Теоретическая информатика изучает математические идеи, лежащие в основе технологий обработки информации. Первый модуль курса посвящён теории формальных языков — разделу теоретической информатики, изучающему конечные автоматы — модель вычислений с конечным объёмым памяти, не зависящим от длины входных данны — и формальные грамматики — модель синтаксиса языков, естественных и искусственных, и подобных им структур. Конечные автоматы используются для формализации простых программ и устройств, а также на их примере будут введены важные понятия недетерминированных и вероятностных вычислений. Будет изучены выразительные возможности разных моделей и построены алгоритмы для анализа их свойств. Наконец, будет формализовано понятие алгоритмически разрешимой задачи и доказана неразрешимость основных задач анализа программ.

На практических занятиях будут решаться математические задачи по изученным на лекциях моделям.

Пререквизиты

- Дискретная математика 1 и 2: логика (предикаты, отношения, кванторы), графы, булевы функции.
- Алгоритмы и структуры данных 1: общая алгоритмическая культура.

Где понадобится

- Во всех последующих математических дисциплинах.
- Для построения алгоритмов и эффективных программ.
- Для понимания дальнейших курсов по теоретической информатике.

Содержание

- Вычислимость с конечной памятью, регулярные языки.
- Конечные автоматы: детерминированные, недетерминированные, двухсторонние, вероятностные.
- Формальные грамматики и алгоритмы синтаксического анализа.
- Вычислимость и неразрешимые задачи, машины Тьюринга.

Вы научитесь

- знать основные виды конечных автоматов, их возможности и ограничения;
- уметь описывать синтаксис языков формальными грамматиками и применять алгоритмы синтаксического анализа;
- знать математическое определение вычислимости и основные алгоритмически неразрешимые задачи.

Теоретическая информатика 1: сложность вычислений

II курс, ноябрь–декабрь.
2 зачётные единицы, 2 часа лекций, 2 часа практик в неделю.
зачёт, экзамен.

Аннотация

Теория сложности вычислений задаёт язык, на котором разговаривают об алгоритмах и криптографических конструкциях. Цель данного курса — ввести основные классы и понятия сложности вычислений (P, NP, сводимости, трудность и полнота, полиномиальная иерархия, полиномиальные схемы, RP и BPP) и доказать несложные соотношения между ними. Уже в самом начале курса мы познакомимся с одной из центральных задач теоретической информатики — вопросе о равенстве классов P и NP, то есть о том, верно ли, что мы можем быстро решить любую задачу, ответы к которой можем быстро проверить.

Практика по курсу состоит в решении математических задач, продолжающих теорию.

Пререквизиты

- Математический анализ 1 и 2: пределы, асимптотики.
- Алгоритмы и структуры данных 1: общая алгоритмическая культура (желательно, но необязательно, иметь представление о вероятностных алгоритмах).
- Формальные языки: конечные автоматы, машина Тьюринга.
- Дискретная математика 1 и 2: логика (предикаты, отношения, кванторы), дискретная теория вероятностей (неравенства Маркова и Чебышева).

Где понадобится

- Во всех последующих математических дисциплинах.
- Для построения и анализа алгоритмов и эффективных программ.
- Для понимания дальнейших курсов по теоретической информатике (теории сложности вычислений и доказательств, криптографии).

Содержание

- Сложность и иерархии по времени и памяти.
- Сводимости и полные задачи. P, NP, полиномиальная иерархия, PSPACE.
- Полиномиальные схемы и параллельные алгоритмы.
- Вероятностные вычисления.

Вы научитесь

- понимать понятия сводимости и полноты;
- ориентироваться в классах сложности вычислений и их взаимоотношениях.

Функциональное программирование

II курс, сентябрь–декабрь.
4 зачётные единицы, 2 часа лекций, 2 часа практик в неделю.
зачёт, экзамен.

Аннотация

Курс знакомит студентов с функциональными языками программирования и методами программирования на этих языках. Рассматриваются отличия функционального подхода к программированию от традиционного императивного, сравниваются их сильные и слабые стороны.

Курс разделен на теоретическую и практическую части. В теоретической части студенты изучат синтаксис и семантику лямбда-исчисления в бестиповом и просто типизированном вариантах. Здесь же они знакомятся с устройством систем типов функциональных языков и, в частности, с алгоритмом вывода типов Хиндли-Милнера.

Практическая часть курса сосредоточена на изучении языка программирования Haskell. Студенты изучат ленивую и энергичную версии операционной семантики, алгебраические типы данных и их использование для реализации механизма сопоставления с образцом. При изучении системы типов языка Haskell будут обсуждаться параметрический и специальный полиморфизм и, в частности, механизм классов типов, в том числе многопараметрических. Подробно рассматриваются основные классы типов из стандартной библиотеки Haskell, студенты приобретают навык программирования с использованием стандартных монад.

Пререквизиты

- Дискретная математика 1: доказательства по индукции.
- Программирование: навык программирования на каком-либо императивном языке.

Где понадобится

- В курсе математической логики в информатике.
- В спецкурсах по верификации ПО, системам типов и системам проверки доказательств.
- В работе на должности разработчика программного обеспечения.

Содержание

- Лямбда-исчисление: термы, редукция, нормальная форма, стратегии редукции, теорема Черча-Россера, теорема о неподвижной точке и рекурсия.
- Типы: простая система типов, версии Карри и Черча, проверка, вывод, обитаемость, алгоритм Хиндли-Милнера.
- Язык Haskell: нестрогая семантика, алгебраические типы данных, сопоставление с образцом; полугруппы и моноиды; свертки и траверсы; функторы, аппликативные функторы и монады; трансформеры монад.

Вы научитесь

- писать код, используя функциональные идиомы;
- разрабатывать программное обеспечения на языке Haskell;
- единообразно рассматривать различные вычислительные эффекты и комбинировать их;
- пользоваться преимуществами ленивой модели исполнения и механизма сопоставления с образцом.

Алгоритмы и структуры данных 2

II курс, сентябрь–декабрь.
5 зачётных единицы, 2 часа лекций, 2 часа практик в неделю.
зачёт, экзамен.

Аннотация

Во второй части курса студенты увидят примеры задач, для которых неизвестно эффективных решений (NP-полные задачи), а также узнают, как эти задачи связаны друг с другом; изучат, как решать эти задачи быстро, но неточно (приближённые алгоритмы); научатся быстро проверять большие числа на простоту, а также изучат основы криптографии — науки о методах шифрования информации; познакомятся с продвинутыми структурами данных, многие из которых используются в базах данных, планировщиках задач в операционных системах и других приложениях.

Пререквизиты

- Алгоритмы и структуры данных 1 и 2: базовые алгоритмы и структуры данных.
- Алгебра 1: основы теории чисел и теории групп, подгруппа, порядок элемента, китайская теорема об остатках.
- Программирование на языке C++: базовое владение языком программирования C/C++.

Где понадобится

- Для написания эффективного и надёжного кода.
- Для прохождения технических интервью.
- В работе на должности разработчика программного обеспечения.

Содержание

- Теория сложности вычислений: классы сложности P и NP, примеры NP-полных задач и их сведения друг к другу.
- Приближённые алгоритмы и алгоритмы кэширования.
- Теоретико-числовые алгоритмы: арифметика сравнений, тест Миллера-Рабина, ρ-алгоритм Полларда, введение в криптографию.
- Продвинутые структуры данных: дерево отрезков, двоичные деревья поиска, skip-листы.

Вы научитесь

- понимать, в каких случаях при решении задачи на практике не стоит рассчитывать на нахождение точного решения;
- реализовывать и применять изученные алгоритмы и структуры данных.

Архитектура компьютера

II курс, сентябрь–декабрь.
4 зачётные единицы, 2 часа лекций, 2 часа практик в неделю.
зачёт, экзамен.

Аннотация

Дисциплина является необходимой для понимания того, как работает компьютер, и знакомит студентов с его организацией сразу на нескольких уровнях. Начиная с электрических сигналов и физического устройства основных узлов компьютера, материал курса постепенно погружает обучающегося в механизмы работы процессора, памяти и в способы коммуникации электронных компонент. Системно рассматриваются такие вопросы, как организация вычислений, устройство памяти на всех уровнях, механизмы прерываний и ввода-вывода. После окончания курса студенты будут уверенно разбираться в том, как работает программное обеспечение на физической аппаратуре. Практические занятия дают возможность закрепить полученные знания путем написания программ на языках `C` и `assembler`, что позволяет студентам максимально сблизиться с низкоуровневым устройством компьютера.

Пререквизиты

- Основы информатики.
- Школьный курс физики.

Где понадобится

- Курс является фундаментальным и пригодится для более глубокого понимания того, как работают современные аппаратные средства, а также для освоения курса Операционные системы.

Содержание

- Введение. История. Электронные компоненты. Управление проводимостью.
- Последовательная и комбинационная логика. Логические схемы.
- Строительные блоки процессора.
- Однотактный процессор.
- Многотактный процессор.
- Проектирование устройства управления.
- Конвейер.
- Организация памяти.
- Кодирование инструкций.
- Прерывания и ввод-вывод.

Вы научитесь

- проектировать процессоры и другие электронные схемы;
- программировать процессоры и контроллеры на `C` и `assembler`.

Базы данных

II курс, сентябрь–декабрь.
4 зачётные единицы, 2 часа лекций, 2 часа практик в неделю.
зачёт.

Аннотация

Курс знакомит студентов с теоретическими основами и практическими методами работы с базами данных. В рамках курса будут рассмотрены правила проектирования и нормализации баз данных. Студенты научатся описывать объекты баз данных, составлять запросы на языке SQL, использовать представления, функции и триггеры, создавать индексы, управлять конкурентным доступом к данным и манипулировать механизмом транзакций.

Большая часть курса посвящена изучению и применению языка SQL, который является стандартом для управления и обработки данных в современных базах данных.

Пререквизиты

- Основы программирования.
- Алгоритмы и структуры данных 1.
- Дискретная математика 1.

Где понадобится

- В работе на должности разработчика программного обеспечения.
- В областях, связанных с работой с данными, таких как Business Intelligence или Data Science.

Содержание

- Проектирование и нормализация реляционных баз данных.
- Основы языка запросов SQL.
- Расширенные возможности SQL.
- Работа с объектами баз данных: функции, представления, триггеры.
- Оптимизация запросов.

Вы научитесь

- проектировать, создавать и оптимизировать реляционные базы данных;
- использовать язык запросов SQL для построения запросов и манипулирования данными;
- создавать программы (скрипты) для реализации многооператорных запросов для обработки данных с помощью языка SQL.

Математический анализ 4

II курс, февраль–май.
5 зачётных единиц, 2 часа лекций, 3 часа практик в неделю.
зачёт, экзамен.

Аннотация

Четвёртая часть курса математического анализа начинается с изучения теории функций одной комплексной переменной (ТФКП). Мы изучим базовые свойства аналитических функций: докажем интегральную формулу Коши, выведем принцип максимума и теорему Руше, изучим различные приложения этих теорем: например, докажем основную теорему алгебры. Будут подробно обсуждаться конформные преобразования и вычисление различных интегралов методом контурного интегрирования.

ТФКП — одновременно красивый и полезный раздел математики. Почти любое утверждение из ТФКП можно превратить в математическую игру: как конформно отобразить одну картинку в другую? Как посчитать сумму обратных квадратов с помощью теоремы о вычетах? В то же время, навыки решения таких головоломок оказываются чрезвычайно полезны: например, конформная инвариантность играет важную роль в современной физике, многие важные асимптотики могут быть вычислены с помощью контурного интегрирования (метод Лапласа).

Во второй половине семестра студенты знакомятся с основами анализа Фурье. Мы начнём с общих теорем о пространствах Лебега и ортогональных системах в Гильбертовых пространствах, а затем перейдём к тригонометрическим рядам Фурье. Помимо основных определений, мы обсудим, когда можно гарантировать сходимость ряда Фурье к значениям исходной функции, как оценить скорость сходимости и асимптотику коэффициентов, какие эффекты возникают в зависимости от характера гладкости функции. Также мы разберём некоторые классические примеры.

Пререквизиты

- Математический анализ 1, 2 и 3.
- Алгебра 1.

Где понадобится

- Для понимания некоторых физических идей, постоянно использующихся на практике.
- Для понимания курса Теория вероятностей.
- Для создания общей математической культуры.

Содержание

- Теория функций комплексной переменной: голоморфность и равносильные ей свойства, интегральная формула Коши, теорема о среднем и принцип максимума, нули и полюсы голоморфной функции, ряды Лорана, теорема Коши о вычетах, вычисление интегралов и сумм рядов с помощью вычетов, локализация корней, применение ТФКП к исследованию производящих функций, конформные отображения.
- Ряды Фурье: пространства Лебега, гильбертовы пространства, коэффициенты Фурье по ортогональной системе, неравенство Бесселя, наилучшие приближения, сходимость тригонометрических рядов Фурье, эффект Гиббса, приближение непрерывных функций многочленами, преобразование Фурье.

Вы научитесь

- применять стандартные теоремы ТФКП для вычислений: вычислять интегралы с помощью теоремы Коши о вычетах, вычислять формулы конформных преобразований, оценивать число решений некоторых уравнений;
- основам теории рядов Фурье: будете знать, откуда берутся формулы для коэффициентов, как оценить асимптотику коэффициентов исходя из априорных свойств функции, в каких случаях ряд Фурье сходится и насколько быстро.

Теория вероятностей

II курс, февраль–май.
4 зачётные единицы, 2 часа лекций, 2 часа практик в неделю.
зачёт, экзамен.

Аннотация

Теория вероятностей — это раздел математики, изучающий закономерности случайных явлений: случайные события, случайные величины, их свойства и операции над ними. Она является основой для других областей математики, таких как математическая статистика, случайные процессы, теория оптимизации, теория игр и финансовая математика. В этом курсе будут рассмотрены как дискретные модели, уже встречавшиеся на курсах дискретной математики, так и непрерывные. В курсе даются базовые понятия теории вероятностей — случайные события и расчет вероятностей, случайные величины, способы их задания и нахождение различных числовых характеристик, основные законы распределения случайных величин, их свойства и применение, производящие и характеристические функции, многомерные случайные величины, цепи Маркова, случайные блуждания. Будут рассмотрены предельные теоремы для сумм случайных величин, закон больших чисел и другие классические результаты теории вероятностей.

Пререквизиты

- Дискретная математика 1: дискретная теория вероятностей.
- Математический анализ 3: теория меры, интегрирование
- Математический анализ 4: ряд Лорана.

Где понадобится

- Для понимания курсов по математической статистике и машинному обучению.
- В таких областях, как машинное обучение и анализ данных.

Содержание

- Случайное событие, условная вероятность, независимость, схема Бернулли, предельные теоремы.
- Случайные величины: плотность и независимость, мат. ожидание и дисперсия, закон больших чисел.
- Метод характеристических функций: обращения, различные виды сходимости случайных величин, центральная предельная теорема.
- Дискретные случайные процессы: марковские цепи, случайные блуждания, ветвящиеся процессы.

Вы научитесь

- находить вероятности событий, описываемых случайными экспериментами;
- работать с вероятностными распределениями, которые регулярно используются на практике;
- вычислять характеристики распределений случайных величин: математическое ожидание, дисперсию, старшие моменты, характеристическую функцию;
- использовать закон больших чисел, центральную предельную теорему и другие классические результаты теории вероятностей;
- работать с цепями Маркова.

Математическая логика в информатике

II курс, февраль–май.
4 зачётные единицы, 2 часа лекций, 2 часа практик в неделю.
экзамен.

Аннотация

Курсы математической логики традиционно ориентированы на подготовку математиков и логиков. Они обычно содержат громоздкие доказательства безусловно важных для логики и формальных оснований математики фактов, но, к сожалению, зачастую не имеющих приложений в других областях. Отбор материала для настоящего курса логики произведён по иным принципам: его содержание исходит из потребностей разработки программного обеспечения (в первую очередь из задачи верификации программ) и включает только самые необходимые разделы математической логики, имеющие непосредственное применение на практике.

Для оживления изложения теоретического материала и повышения вовлечённости студентов в курсе используется целый ряд программных инструментов, таких как система интерактивного доказательства теорем Lean, SMT-решатель Z3, языки программирования Prolog и Haskell, и некоторые другие. Разумеется, весь этот инструментарий не отменяет столь любимого студентами построения формальных логических доказательств с обязательным рисованием деревьев логического вывода на бумаге.

Основная цель курса — подготовить студентов к профессиональному использованию аппарата математической логики в научных исследованиях и при разработке программного обеспечения в отраслях, где корректность кода оказывается критически важной.

Пререквизиты

- Дискретная математика 1: множества, доказательства, в том числе по индукции.
- Дискретная математика 2: модели вычислений и классы сложности.
- Функциональное программирование: лямбда-исчисление, простая система типов, язык Haskell.

Где понадобится

- В курсах по теории языков программирования (теории типов) и верификации программного обеспечения.
- Для применения формальных методов к информационным технологиям.
- В научных исследованиях в области теоретической информатики.

Содержание

- Исчисление высказываний.
- Логика первого порядка (исчисление предикатов).
- Классическая и интуиционистская логики.
- Элементы теории моделей и теории доказательств.
- Язык Prolog и логическое программирование.
- SMT-решатели и их приложения.
- Соответствие Карри–Ховарда и методы верификации ПО на его основе.
- Темпоральные логики и проверка моделей.
- Модальные логики и мультиагентные системы.

Вы научитесь

- формализовывать рассуждения относительно сущностей реального мира;
- использовать системы автоматического и интерактивного доказательства теорем;
- решать задачи методами логического программирования;
- применять SMT-решатели в программных системах;
- использовать различные инструменты верификации программного обеспечения.

Теоретическая информатика 2: вычислимость

II курс, февраль–март.
2 зачётные единицы, 2 часа лекций, 2 часа практики в неделю.
зачёт, экзамен.

Аннотация

Теоретическая информатика изучает математические идеи, лежащие в основе технологий обработки информации. Третий модуль курса посвящён теории вычислимости — разделу на стыке теоретической информатики, математической логики и теории алгоритмов, возникшему в результате изучения понятий вычислимости и невычислимости. В модуле изучаются различные модели вычислимости и доказывается их эквивалентность (машины Тьюринга, рекурсивные функции), доказывается алгоритмическая неразрешимость ряда задач, доказываются некоторые классические теоремы теории вычислимости (теорема Райса и теорема о неподвижной точке, а также её следствие о программе, печатающей свой текст), вводится понятие арифметической иерархии.

Практика по курсу состоит в решении математических задач, продолжающих теорию.

Пререквизиты

- Дискретная математика 1 и 2: логика (предикаты, отношения, кванторы), графы, булевы функции.
- Алгоритмы и структуры данных 1: общая алгоритмическая культура.

Где понадобится

- Во всех последующих математических дисциплинах.
- Для построения алгоритмов и эффективных программ.
- Для понимания дальнейших курсов по теоретической информатике.

Содержание

- Вычислимые функции, разрешимые и перечислимые множества.
- Универсальные нумерации. Теорема Райса. Теорема о неподвижной точке. Программа, печатающая свой текст.
- Алгоритмически неразрешимые задачи: проблема остановки, проблема соответствия Поста, проблема домино.
- Сводимость и арифметическая иерархия, полные множества в классах иерархии.

Вы научитесь

- знать и понимать различные формализации понятия алгоритма;
- знать примеры алгоритмически неразрешимых задач и уметь ими пользоваться для доказательства неразрешимости;
- понимать классы арифметической иерархии.

Теоретическая информатика 2: теория информации

II курс, апрель–май.
2 зачётные единиц, 2 часа лекций, 2 часа практик в неделю.
зачёт, экзамен.

Аннотация

Часто у людей есть какое-то интуитивное понятие «количества информации». Например в книге «Война и мир» содержится больше информации, чем, к примеру, на этикетке от сока. В данном курсе мы формализуем это интуитивное представление, что даст нам полезный математический инструмент, который поможет, в частности, понять, какие из вычислительных задач являются простыми, а какие сложными. Пользуясь теорией информации мы покажем: как можно эффективно закодировать данные; как разделить секрет между несколькими людьми таким образом, чтобы они смогли его узнать, только в случае если они соберутся вместе.

Одной из основных частей курса является «коммуникационная сложность», где мы попытаемся ответить на вопрос: каким количеством информации два человека должны обменяться, чтобы совместно решить какую-нибудь задачу. Коммуникационная сложность — один из активно развивающихся разделов теоретической информатики, применяющийся во всех областях, и в частности, для анализа эффективности алгоритмов. Мы рассмотрим, как классические подходы к данной проблеме, так и современный теоретико-информационный подход.

Ближе к концу курса мы рассмотрим «Колмогоровскую сложность», где мы изучим свойства случайных объектов и поймем, какие именно объекты мы можем считать случайными, а какие нет.

Пререквизиты

- Дискретная математика 1: дискретная теория вероятностей.
- Дискретная математика 2: основы теории графов.
- Алгебра 1: линейная алгебра, многочлены.
- Теоретическая информатика 1, теория сложности.
- Теоретическая информатика 2: вычислимость.

Где понадобится

- Во всех последующих дисциплинах по теоретической информатике.
- Для построения и анализа алгоритмов.
- Для понимания теории кодирования.

Содержание

- Понятие информации. Информация по Хартли.
- Энтропия распределения. Информация по Шеннону.
- Энтропийная оценка на биномиальные коэффициенты.
- Однозначно декодируемые коды. Неравенство Крафта–Макмиллана.
- Код Шеннона–Фано, код Хаффмана, арифметическое кодирование.
- Криптографические протоколы: задача о разделении секрета.
- Коммуникационная сложность. Оценки на коммуникационные протоколы.
- Игра Карчмера–Вигдерсона.
- Применение теории информации в коммуникационной сложности. Теорема Храпченко.
- Колмогоровская сложность.
- Случайность по Мартин-Лёфу.

Вы научитесь

- оценивать «количество информации» во множествах;
- эффективно кодировать данные;
- использовать понятие «информации» для получения оценок на время работы алгоритмов;
- строить и оценивать коммуникационные протоколы;
- применять идеи алгебры и дискретной математики для решения теоретических задач.

Алгоритмы и структуры данных 3

II курс, февраль–май.
4 зачётные единицы, 2 часа лекций, 2 часа практик в неделю.
зачёт, экзамен.

Аннотация

В третьей части курса студенты научатся быстро осуществлять арифметические операции с длинными числами; изучат продвинутые алгоритмы на графах (паросочетания, потоки), а также алгоритмы и структуры данных, предназначенные для работы со строками.

Пререквизиты

- Алгоритмы и структуры данных 1 и 2: базовые алгоритмы и структуры данных.
- Алгебра 1: комплексные числа.
- Программирование на языке C++: базовое владение языком программирования C/C++.

Где понадобится

- Для написания эффективного и надёжного кода.
- Для прохождения технических интервью.
- В работе на должности разработчика программного обеспечения.

Содержание

- Быстрое преобразование Фурье, быстрое деление методом Ньютона.
- Паросочетания: алгоритм Куна, алгоритм поиска стабильного паросочетания.
- Потоки и разрезы: алгоритмы поиска максимального потока, минимального глобального разреза.
- Алгоритмы на строках: поиск подстроки в строке, алгоритм Ахо-Корасик, суффиксные структуры данных.

Вы научитесь

- реализовывать и применять изученные алгоритмы и структуры данных.

Операционные системы

II курс, февраль–май.
4 зачётные единицы, 2 часа лекций, 1 час практик в неделю.
зачёт, экзамен.

Аннотация

Курс дает системный взгляд на принципы работы операционных систем и их архитектуру. В рамках данного курса рассматривается история и основные этапы развития операционных систем, общие принципы организации современных операционных систем общего назначения, объясняется, каким образом сложилась современная архитектура. Основное внимание уделено механизмам планирования исполнения процессов и потоков, организации памяти, средствам межпроцессного взаимодействия. Обзорно рассматриваются особенности мобильных операционных систем и систем реального времени. Ключевым аспектом курса являются практические работы, в рамках которых студенты получают практические навыки разработки реальных ОС.

Пререквизиты

- Основы Linux.
- Низкоуровневое программирование на языке C.
- Архитектура ЭВМ: знание принципов работы современных аппаратных средств.

Где понадобится

- Курс является фундаментальным и пригодится для более глубокого понимания дисциплин, связанных с разработкой программного обеспечения, особенно, при разработке серверных и мобильных приложений и системного программного обеспечения.
- Для понимания курса Параллельное программирование.

Содержание

- Введение. Генезис операционных систем.
- Основные абстракции ядра ОС.
- Процессы и потоки.
- Загрузка ОС и создание первого процесса.
- Планирование и многозадачность.
- Примитивы синхронизации.
- Управление памятью и защищённый режим.
- Файловые системы.
- Управление пользовательским окружением.
- Реализация межпроцессного взаимодействия.

Вы научитесь

- разбираться в устройстве операционных систем;
- программировать основные компоненты ОС, такие как планировщик процессов, менеджер памяти, обработчики прерываний ввода-вывода;
- писать драйверы.

Математическая статистика

III курс, сентябрь–декабрь.
4 зачётные единицы, 2 часа лекций, 2 часа практик в неделю.
зачёт, экзамен.

Аннотация

Математическая статистика — это раздел математики, который занимается описанием и анализом данных для построения вероятностных моделей случайных явлений, эти данные породивших. Описанное промежуточное положение между реальными наблюдениями и абстрактными вероятностными моделями делает математическую статистику основным инструментом решения прикладных задач, в которых неопределённость интерпретируется как случайность. В этом курсе мы познакомимся как с классическими результатами в этой области, так и с более современными вычислительными методами. Практические занятия будут проводиться на Python, на них мы будем применять полученные знания на реальных данных и обозначим границы применимости изученных методов.

Пререквизиты

- Теория вероятностей: основы, случайные величины, предельные теоремы.
- Математический анализ 3: теория меры, интегрирование.
- Алгебра 2: линейная алгебра, ортогональность.
- Программирование: базовое владение языком программирования Python.

Где понадобится

- Для понимания курса Машинное обучение.
- В работе на любой должности, связанной с прогнозированием и аналитикой.
- В жизни для более критического и осмысленного восприятия информации.

Содержание

- Основы математической статистики: описательная статистика, оценка параметров, доверительные интервалы, проверка гипотез, критерии согласия и однородности, линейная регрессия.
- Метод Монте-Карло, методы ресемплирования.
- Основы байесовского подхода, байесовская классификация.

Вы научитесь

- визуализировать данные для проведения предварительного анализа;
- оценивать неизвестные параметры исследуемой модели и определять качество полученных оценок;
- формулировать задачи в терминах статистических гипотез и проверять их;
- использовать бутстрэп для оценки параметров и проверки гипотез;
- работать с линейными моделями, определять их качество и использовать их для прогнозирования;
- использовать пакеты `scipy.stats`, `pandas`, `matplotlib`, `sklearn` и др. для решения описанных выше задач на реальных данных.

Машинное обучение 1

III курс, сентябрь–декабрь.
4 зачётные единицы, 2 часа лекций, 2 часа практик в неделю.
зачёт, экзамен.

Аннотация

Целью курса является демистификация машинного обучения. В рамках курса предполагается подробное рассмотрение простейших моделей машинного обучения, так как именно такой подход позволяет осознать основные принципы данной области в целом. Знание этих принципов даст студентам возможность самостоятельно понимать механизмы функционирования более сложных современных моделей, обнаруживать пути улучшения уже существующих алгоритмов, а также адаптировать методы машинного обучения для решения нестандартных задач.

Пререквизиты

- Алгебра 1 и 2.
- Теория вероятностей.
- Математическая статистика.
- Математический анализ 1, 2, 3 и 4.
- Программирование: базовое владение языком программирования `Python` (ввод-вывод, циклы, рекурсия, классы, функции).

Где понадобится

- Более узкоспециализированные курсы по машинному обучению.
- В работе на должности Data Scientist, Machine Learning Engineer, Machine Learning Researcher.

Содержание

- Задачи машинного обучения.
- Линейные методы классификации и регрессии.
- Логические методы классификации.
- Метрические методы классификации и регрессии.
- Метрики качества, обобщающая способность.
- Методы отбора признаков.
- Основы байесовских методов.

Вы научитесь

- ориентироваться в обширной области машинного обучения;
- понимать на базовом уровне принципы устройства различных методов;
- формализовывать задачи на языке обучения моделей;
- понимать преимущества и недостатки моделей;
- реализовывать базовые модели на языке программирования `Python`;
- азам использования таких библиотек языка программирования `Python`, как `numpy`, `matplotlib` и `pandas`;
- бороться с проблемами, которые возникают при обучении моделей.

Компьютерные сети

III курс, февраль–май.
4 зачётные единицы, 2 часа лекций, 2 часа практик в неделю.
зачёт, экзамен.

Аннотация

Целью курса является предоставить базовые знания о принципах работы современных компьютерных сетей. Будут рассмотрены все уровни сетевой архитектуры — от физического до прикладного. Разобраны базовые компоненты сети, важнейшие службы и протоколы, принципы взаимодействия сетей друг с другом, беспроводные сети, пиринговые сети, потоковое вещание, интернет-телефония и др. Затронуты особенности аппаратного и программного обеспечения. Также будут рассмотрены мобильные сети 3G/LTE/5G. Отдельное внимание уделено теме сетевой безопасности.

Пререквизиты

- Базовое владение одним из языков программирования (`Python`, `Java`, `C#`).
- Базовое владение на уровне пользователя ОС `Windows`/`Linux`.

Где понадобится

- В курсе по сервисам и микросервисам.
- Для администрирования компьютерных систем.
- В работе на должности разработчика программного обеспечения.

Содержание

- Основы сетей передачи данных.
- Пять уровней Интернета (физический, канальный, сетевой, транспортный и прикладной).
- Беспроводные и мобильные сети.
- Мультимедийные сетевые технологии.
- Безопасность компьютерных сетей.

Вы научитесь

- работать с компьютерными сетями на начальном уровне;
- понимать, как сети устроены изнутри;
- использовать фундаментальные знания для разработки распределенных приложений.

Машинное обучение 2

III курс, февраль–май.
4 зачётные единицы, 2 часа лекций, 2 часа практик в неделю.
зачёт, экзамен.

Аннотация

На практике перед специалистом по машинному обучению часто встают нестандартные задачи. Многие из них трудно, а иногда даже невозможно сформулировать и решить, используя напрямую такие ”классические”методы машинного обучения, как регрессия, классификация и кластеризация. Причин тому может быть очень много: потенциальная потребность в учете бесконечного количества факторов, их многочисленность и примитивность, отсутствие факторов в их привычном понимании, потребность в генерации новых объектов, необходимость поиска причинно-следственных связей, потребность в обучении и использовании алгоритмов при изначальном отсутствии какой-либо выборки, необходимость интерпретации сложных моделей. Целей у данного курса две: во-первых, научить студента адаптировать уже известные алгоритмы машинного обучения для решения подобных задач; во-вторых, познакомить с большим количеством новых методов, изначально спроектированных для их решения.

Пререквизиты

- Машинное обучение 1.

Где понадобится

- Более узкоспециализированные курсы по машинному обучению.
- В работе на должности Data Scientist, Machine Learning Engineer, Machine Learning Researcher.

Содержание

- Временные ряды.
- Глубокие нейронные сети.
- Методы обучения ранжированию.
- Коллаборативная фильтрация и матричные разложения.
- Тематическое моделирование.
- Обучение с подкреплением.
- Активное обучение.
- Causal impact и uplift моделирование.
- Интерпретация моделей машинного обучения.

Вы научитесь

- ориентироваться в обширной области машинного обучения;
- понимать на базовом уровне принципы устройства различных методов;
- формализовывать задачи на языке обучения моделей;
- решать широкий спектр задач с помощью машинного обучения;
- понимать преимущества и недостатки моделей;

Проектирование высоконагруженных систем

IV курс, сентябрь–декабрь.
4 зачётные единицы, 2 часа лекций, 2 часа практик в неделю.
зачёт, экзамен.

Аннотация

Как конструировать из отдельных приложений сложные системы, стоящие за реальными продуктами? Какие сложности и ограничения при этом возникают? Как понять, какие ресурсы понадобятся для системы? Это практический курс по разработке высоконагруженных систем. Он нацелен на то, чтобы учащийся мог самостоятельно сделать готовую программную систему, состоящую из множества компонент. Данный курс содержит в себе: изучение языка Go как одного из самых популярных языков для написания облачных, сетевых и web-сервисов; устройство современных web-сервисов; работу распределенных систем, сервисную архитектуру; культуру и практики DevOps; и поставку распределенных приложений.

Пререквизиты

- Алгоритмы и структуры данных 1.
- Основы программирования.
- Базы данных.
- Основы Linux.

Где понадобится

- В командных проектах.
- В работе на должности разработчика программного обеспечения.

Содержание

- Программирование на Go.
- Web-сервисы.
- Распределенные системы.
- Сервисная архитектура.
- Мониторинг.
- Поставка приложения.

Вы научитесь

- разрабатывать сервисы на Go;
- проектировать распределенные системы под высокую нагрузку;
- производить анализ и мониторинг программных систем.

Программная инженерия

IV курс, сентябрь–декабрь.
4 зачётные единицы, 2 часа лекций, 2 часа практик в неделю.
зачёт, экзамен.

Аннотация

Курс посвящен вопросам командной разработки программных продуктов. Будут рассмотрены основные этапы разработки и весь жизненный цикл приложения. В рамках команд слушатели будут разрабатывать проекты по процессам, приближенным к тем, которые используются в промышленной разработке. В отличие от практики в какой-либо компании, где обычно предполагается участие в уже существующем проекте и работа над отдельными часто второстепенными компонентами, на курсе у вас будет возможность поучаствовать во всех этапах жизненного цикла проекта, включающих сбор требований, проектирование, прототипирование, развертывание и др. Таким образом, вы получите целостное представление о том, как создаются проекты в команде и как работает ваш проект.

На курсе мы поговорим об анализе требований, проектировании и архитектуре программных систем, тестировании, рефакторинге и других темах, важных при разработке. Подробно рассмотрим гибкие методологии разработки. Особое внимание на курсе будет уделено практике командной разработки по Agile. Слушателям будет предложено разбиться на несколько команд и разработать прототип одной из распределенных высоконагруженных систем (например, социальная сеть или сервис онлайн-продаж). Тему проекта каждая из команд сможет выбрать самостоятельно. В ходе работы над проектом будет возможность на практике познакомиться не только с основными составляющими гибких методологий разработки ПО, но также с некоторыми шаблонами корпоративных приложений.

Пререквизиты

- Основы программирования
- ООП и проектирование
- Базы данных

Где понадобится

- В работе на должности разработчика программного обеспечения
- В работе на должности менеджера проектов
- При разработке собственных проектов, в стартапах

Содержание

- Управление программными проектами
- Инструменты и средства командной разработки
- Проектирование и архитектура программных систем
- Принципы построения высоконагруженных распределенных систем
- Сервисы и микросервисная архитектура
- Работа с базами данных, ORM системы
- Юнит тестирование и интеграционное тестирование
- Рефакторинг и чистый код

Вы научитесь

- Работать в командах по Agile
- Использовать современные инструменты командной разработки (Jira, Azure DevOps, Git)
- Проектировать, разрабатывать и интегрировать отдельные компоненты в общий проект
- Настраивать процесс CI/CD

C# и .Net

5–8 семестры (дисциплина по выбору).
3–4 зачётные единицы (в зависимости от отчётности).
зачёт или экзамен.

Аннотация

Курс посвящен языку C#, исполняющей среде CLR и технологиям .Net.

C# - современный динамично развивающийся кросс-платформенный язык программирования, который используется при разработке самых разных программных систем - это корпоративные приложения и облака Azure, игры и Unity, Stackoverflow и др.

C# имеет активную поддержку и отражает современные тенденции развития языков ООП. Поэтому важно познакомиться с тем, какие возможности предлагает C#, а также узнать, как некоторые из этих возможностей реализованы изнутри. Поскольку основные концепции работы языков с виртуальными машинами схожи, вам будет проще изучать другие языки программирования и понимать, как они работают.

На курсе вы познакомитесь с основными возможностями языка C#, а также с особенностями работы CLR, такими как исполнение кода, сборка мусора, отражение и др. Узнаете, во что (и почему) разворачивается ”синтаксический сахар”языка и для чего нужен IL (intermediate language). Разберем, как реализованы некоторые структуры данных изнутри (List, Queue, Dictionary, ConcurrentDictionary). Поговорим о поддержке многопоточности со стороны C#. Рассмотрим поддержку основных концепций ООП, особенности реализации некоторых паттернов проектирования и принципы SOLID.

Кроме того, рассмотрим основные технологии .Net. Вы узнаете, как создавать и разворачивать сервисы, разрабатывать GUI, работать с базами данных из вашей программы, синхронизировать потоки и работать с юнит тестами. Для того, чтобы лучше познакомиться с возможностями .Net, в течение курса вы будете разрабатывать свой программный проект с использованием всех рассмотренных технологий.

Пререквизиты

- Основы программирования
- Алгоритмы и структуры данных

Где понадобится

- В работе на позиции C# разработчика
- В курсе ”Пограммная инженерия”
- В разработке современных высоконагруженных приложений

Содержание

- Язык C#. Основы типов. Делегаты. Строки и исключения. Коллекции.
- CLR. Управление памятью. Исполнение кода, хостинг, отражение
- C# и ООП. SOLID. Паттерны проектирования
- Многопоточность в C#. Примитивы синхронизации. ThreadPool, async\await, TPL
- GUI: WPF. MVVM, MVC
- Работа с базами данных. Ado.Net. EF.
- Сервисы. WebAPI. WCF
- Сетевое программирование в C#
- Юнит тестирование. NUnit, NSubstitute, Moq

Вы научитесь

- Разрабатывать ОО-программы на C#
- Понимать принципы работы виртуальной машины CLR
- Создавать полноценные приложения на основе технологий .Net

Архитектура JVM

5–8 семестры (дисциплина по выбору).
3–4 зачётные единицы (в зависимости от отчётности).
зачёт или экзамен.

Аннотация

Языки на платформе JVM (Java, Kotlin, Scala и другие) широко применяются в индустрии и, несомненно, в некоторой степени изучены студентами. Цель данного курса — дать фундаментальные представления о том, как устроена виртуальная машина, исполняющая скомпилированный код на этих языках.

В курсе даётся общий обзор архитектуры JVM и жизненного цикла программы, а также более подробно рассматриваются определяющие JVM технологии: байт-код, интерпретация/JIT-компиляция, сборка мусора и так далее.

Помимо теоретического представления об устройстве JVM, студенты получат практический опыт написания компонентов виртуальной машины, соответствующей подмножеству официальной спецификации JVM.

Пререквизиты

- Программирование: владение JVM-языком (Java, Kotlin, Scala).

Где понадобится

- При написании кода на JVM-языках.
- При написании виртуальных машин.

Содержание

- Обзор архитектуры JVM.
- Байт-код, классы и методы.
- Модель памяти, вызовы методов.
- Интерпретатор и JIT-компилятор.
- Сборка мусора.

Вы научитесь

- понимать, что происходит при работе JVM;
- понимать производительность JVM;
- искать утечки памяти в JVM;
- отвечать на (некоторые) загадочные вопросы на собеседовании;
- читать спецификацию.

Введение в метавычисления

5–8 семестры (дисциплина по выбору).
3–4 зачётные единицы (в зависимости от отчётности).
зачёт или экзамен.

Аннотация

Курс даёт представление о теоретических и практических аспектах метавычислений — области Computer Science, которая изучает методы преобразования программ. Классическим примером метапрограммы является компилятор: он преобразует программу на одном (как правило, высокоуровневом) языке в программу на другом (как правило, низкоуровневом языке). Теория построения компиляторов существует сама по себе как отдельная область Computer Science. Однако она рассматривает проблему построения компилятора довольно узко, просто указывая на способы программирования отдельных его частей, таких как лексический или синтаксический анализатор. Метавычисления также изучают компиляторы, однако в другом ключе: в совокупности с другими метапрограммами. Каким образом метапрограммы взаимодействуют между собой? Можно ли автоматически построить компилятор для заданного языка программирования? Это именно те вопросы, которые будут отражены в данном курсе.

Первая часть курса будет посвящена построению метапрограмм для простого, специально созданного языка. Вторая часть — вопросам, которые возникают при применении этой теории к реальным языкам. В конце курса мы обсудим, как это все работает в реально существующих проектах.

Пререквизиты

- Алгоритмы и структуры данных.
- Теоретическая информатика.
- Функциональное программирование.

Где понадобится

- Для понимания особенностей реализации интерпретаторов, компиляторов и виртуальных машин.
- Для понимания особенностей дизайна языков программирования и связей между ними.

Содержание

- Введение: языки программирования, интерпретатор и компилятор, проекции Футамуры.
- Проблема суперкомпиляции. Работа с множествами. Дерево процессов. Язык реализации. Семантика языка `TSG` и его интерпретатор.
- Представление множеств. Конфигурационные переменные (`C`-переменные). Конфигурационные связи, среды и состояния. Рестрикции и подстановки. Свойства подстановок.
- Отождествление `C`-выражений. Представляемое множество. Полная подстановка. Классы, `L`-классы.
- Суперпозиция подстановок. Сужения и разбиения множеств. Каноническая форма класса. Надклассы и подклассы.
- Дерево процессов. Построение дерева процессов.
- Окрестностный анализ. Окрестностное тестирование.
- Инверсное программирование, декларативные языки. Нестандартные семантики.
- Суперкомпилятор для языка `TSG`.
- Специализация императивных языков.
- Самоприменение.
- Специализация подмножества `Scheme`.
- Виртуальные машины, архитектура `GraalVM`.
- Использование `Truffle`.

Вы научитесь

- разбираться в устройстве интерпретаторов, виртуальных машин и компиляторов;
- понимать связи между интерпретацией и компиляцией программ;
- реализовывать собственные языки программирования.

Выпуклая оптимизация

5–8 семестры (дисциплина по выбору).
3–4 зачётные единицы (в зависимости от отчётности).
зачёт или экзамен.

Аннотация

Выпуклая оптимизация обобщает теорию линейного и квадратичного программирования. Курс затрагивает основные концепции математической оптимизации и базовые методы для решения различных прикладных задач.

Пререквизиты

- Математический анализ 1, 2, 3 и 4.
- Алгебра 1 и 2.
- Программирование: базовое владение языком программирования `Python`.

Где понадобится

- Подавляющее большинство современных методов машинного обучения основаны на выпуклой оптимизации.
- Большой пласт задач автоматического управления покрывается выпуклой оптимизации.
- Является одним из основных математических инструментов в экономике.

Содержание

- Квадратичные функции.
- Метод множителей Лагранжа и его обобщения.
- Основные разновидности градиентного спуска.
- Метод Ньютона и метод внутренней точки.

Вы научитесь

- реализовывать и оценивать сложность основных методов оптимизации;
- сводить некоторые практические задачи к задачам оптимизации;
- подбирать наиболее подходящий алгоритм оптимизации к конкретной задаче.

Компьютерная графика

5–8 семестры (дисциплина по выбору).
3–4 зачётные единицы (в зависимости от отчётности).
зачёт или экзамен.

Аннотация

Данный курс познакомит с основными понятиями и алгоритмами классической трёхмерной компьютерной графики и покажет, как их реализовать на GPU с использованием OpenGL API. Вы научитесь эффективному рендерингу сложных трёхмерных объектов и сцен, познакомитесь с деталями реализации многих графических эффектов, узнаете о подробностях работы современных графических движков. Полученные знания будут полезны как при работе с существующими движками, так и при реализации собственного рендеринга, и могут быть применены в большом кругу задач, включающем игровую трёхмерную графику, трёхмерную анимацию, или научную визуализацию.

Пререквизиты

- Алгебра 1 и 2.
- Дискретная математика 1.
- Основы программирования.
- Программирование на языке C++.

Где понадобится

- Работа с графическими движками и их реализация
- Эффективная визуализация данных.
- Научная визуализация.

Содержание

- История, назначение и различия графических API.
- Ортографическая и перспективная проекции, аффинные преобразования, однородные коодинаты, кватернионы.
- Графический конвейер: primitive assembly, шейдеры, растеризация.
- Основные примитивы OpenGL: буферы, VAO, шейдеры, текстуры.
- Виды и опции рендерига: индексировнный рендериг, instancing, тест глубины, stencil test, отсечение невидимых поверхностей, полупрозрачность.
- Текстуры: виды, опции фильтрации, mipmaps.
- Освещение: модели освещения, виды источников света, normal/bump/material mapping, baking.
- Рендеринг в текстуру и постобработка: depth blur, SSAO, гамма-коррекция, сглаживание.
- Тени: shadow volumes, shadow mapping, мягкие тени.
- Deferred rendering.
- Оптимизация рендеринга: batching, LOD, frustum culling, occlusion culling.

Вы научитесь

- использовать OpenGL API;
- реализовывать различные алгоритмы компьютерной графики;
- реализовывать компоненты графических движков.

Компьютерное зрение

5–8 семестры (дисциплина по выбору).
3–4 зачётные единицы (в зависимости от отчётности).
зачёт или экзамен.

Аннотация

В курсе по комьютерному зрению вы познакомитесь с тем, как с помощью компьютера можно интерпретировать различные визуальные данные. Мы пройдем полный путь изучения алгоритмов, которые применяются с момента, как на ПЗС матрицу камеры накапливается сигнал, до момента, когда мы, например, сможем найти треки объектов на видео, которое записывает эта камера. Уделим основное внимание тому, как извлечь из изображеня семантическию информацию разного уровня сложности. Рассмотрим эвристические алгоритмы, с помощью которых можно строить вычислительноэффективные интерпретируемые решения полезные на начальном этапе решения бизнес задач связанных с анализом изображений. Разберемся с самыми разными гистрограммами и поиском ключевых точек. Проложим мостик от классических алгоритмов машинного обучния (Деревья, SVM, Градиентный бустинг) к глубоким нейронным сетям. Примение глубоких сетей в компьютерном зрении – стандарт современной индутрии. Большинство задач поиска, дектекции и классификации решается с помощью нейросетей. Мы с вами тоже прорешаем эти задачи: рассмотрим данные архитектуры и метрики, которые используются.

Пререквизиты

- Программирование на python.
- Машинное обучение.

Где понадобится

- В работе на должности Data Scientist, Computer Vision Engineer, Computer Vision Researcher.

Содержание

- Базовая обработка изображений. Работа с цветовыми пространствами.
- Обработка изображений с помощью фильтров. Физический смысл фильтров. Преобразование Фурье.
- Локальные и глобальные эвристические признаки для изображений.
- Параметрические модели: преобразование Хафа, RANSAC
- Классификация изображерний.
- Нейросети для изображений. Базовые слои и архитектуры.
- Детектривание объектов: от Viola Jones до YOLO.
- Сегментация изображений. От GraphCut до Mask-RCNN.
- Генеративные сети для изображерний: Style Transfer, CycleGAN etc
- Цифровой фотомонтаж.
- Извлечение фона, оптический поток.

Вы научитесь

- Строить эвристические алгоритмы для протейших бейзлайнов.
- Оценивать, как качество и скорость работы алгоритмов зависит от данных, моделей, фреймворков, вычислительных компонент.

Методы и алгоритмы эвристического поиска

5–8 семестры (дисциплина по выбору).
3–4 зачётные единицы (в зависимости от отчётности).
зачёт или экзамен.

Аннотация

Эвристический поиск в пространстве состояний – один из основных инструментов искусственного интеллекта при решении сложных задач, таких как планирование траектории мобильных роботов, игра в Го, синтез новых лекарств и так далее. Он может использоваться как самостоятельно, так и в совокупности с другими методами, например, методами машинного обучения, которые “выучивают” домено-зависимые эвристики для повышения эффективности решения поставленных задач.

Основная цель курса – овладеть различными методами и алгоритмами эвристического поиска и научиться применять их для решения нетривиальных задач, для которых классические методы перебора оказываются неэффективными.

По всем основным темам курса предусмотрены лабораторные работы на Python. Итоговое испытание (экзамен) – защита индивидуального или группового программного проекта.

Пререквизиты

- Математика: школьная программа, основы теории графов (основные определения и базовые алгоритмы).
- Программирование: знание базовых алгоритмов и структур данных (стек, очередь, список, словарь и др.), базовое владение языком программирования Python (ввод-вывод, циклы, рекурсия).

Где понадобится

- При решении задач как альтернатива методам полного перебора.
- Для лучшего понимания курсов по икусственному интеллекту и машинному обучению.
- При разработке программных систем навигации, маршрутизации, планирования и др.

Содержание

- Методы перебора для неинформированного поиска: BFS, DFS, DFID.
- Алгоритм A*: принцип работы, понятие эвристической функции, свойства алгоритма.
- Эвристики: свойства, взаимосвязь со свойствами алгоритма поиска.
- Модификации алгоритма A*: поиск с итерационным заглубением (IDA*), поиск суб-оптимальных решений (WA*, FocalA*), двунаправленный поиск и др.
- Применения алгоритмов эвристического поиска для решения задач планирования.
- Методы поиска для игр с двумя игроками.

Вы научитесь

- формулировать сложные задачи на языке “поиска в пространстве состояний”;
- различным подходам к организации информированного поиска;
- реализовывать на практике алгоритмы эвристического поиска и применять из для решения различных задач;
- проводить самостоятельные исследования разрабатываемых алгоритмов.

Параллельное программирование

5–8 семестры (дисциплина по выбору).
3–4 зачётные единицы (в зависимости от отчётности).
зачёт или экзамен.

Аннотация

Основная цель курса — обоснованно выбирать средства достижения необходимого уровня производительности программ и квалифицированно эти средства применять. Курс носит больше прикладной, нежели теоретический характер и обновляется в соответствии с текущими общемировыми практиками разработки. В то же время, много внимания будет уделено «классическим» вопросам параллельного программирования:

- синхронизации;
- потокам и процессам;
- архитектурным шаблонам;
- разработке потокобезопасных структур данных.

Технологии, познаваемые в рамках курса, в том числе закрепляемые на практических занятиях, варьируются от наиболее низкоуровневых подходов — потоков ОС — к постепенно повышающим уровень абстракции, вплоть до сопрограмм и транзакционной памяти. Мы постараемся выработать объективные критерии применимости конкретных технологий, подходов и способов их применения в практических задачах.

Курс содержит практические задания с использованием различных языков программирования, призванных научить студентов «параллельному» мышлению в ходе разработки и отладки многопоточного кода.

Пререквизиты

- Программирование на языке C (средний уровень).
- Java или любой язык на JVM (средний уровень).
- Операционные системы.
- Технологии разработки ПО (начальный уровень).
- Архитектура ПО (начальный уровень).

Где понадобится

- В работе разработчика.
- Для более эффективной реализации программ.

Содержание

- Процессы/потоки: критерии выбора архитектуры, жизнь в ядре ОС.
- Синхронизация: примитивы, алгоритмы, lock-free, wait-free, транзакционная память, консенсус.
- Шаблоны параллельного программирования: базовые шаблоны, асинхронный ввод/вывод.
- Иные технологии повышения производительности: компиляторы, SSE, OpenCL.
- Профилирование и поиск ошибок.
- Технологии: потоки ОС, OpenMP, Intel TBB, Java.util.concurrent, Coroutines.

Вы научитесь

- ориентироваться в применимости различных технологий под конкретные задачи;
- осознанно выбирать способы передачи данных между параллельными потоками/процессами;
- проектировать неизбыточную архитектуру для заданного уровня производительности;
- понимать ограничения согласованности и актуальности данных;
- знать низкоуровневое устройство рассматриваемых механизмов.

Программирование в Linux

5–8 семестры (дисциплина по выбору).
3–4 зачётные единицы (в зависимости от отчётности).
зачёт или экзамен.

Аннотация

Основные цели курса:

- познакомить с системным программированием на примере программирования в Linux;
- заложить навыки написания системного ПО — низкоуровневого кода, непосредственно взаимодействующий с ресурсами операционной системы;
- дать представление о существующих средствах анализа взаимодействия программ с операционной системой.

Пререквизиты

- Программирование на языке C++.
- Основы Linux: навыки пользователя.
- Архитектура ЭВМ: организация памяти.

Где понадобится

- В работе на должности системного программиста.
- В работе в сфере кибербезопасности.
- Для более глубокого понимания взаимодействия программ и операционной системы.
- При программировании встраиваемых систем (embedded systems).

Содержание

- Загрузка и исполнение программ: executable and linkable format, разделяемые библиотеки, механизмы загрузки, системные вызовы.
- Всё есть файл: виды файлов, файловые системы и операции над файлами, виртуальные файловые системы.
- Процессы и потоки: иерархия, межпроцессное взаимодействие, планировщик, ограничение ресурсов, сигналы, POSIX-threads.
- Модули ядра: виртуальная и физическая память, драйверы устройств.

Вы научитесь

- эффективно использовать механизм разделяемых библиотек;
- исследовать взаимодействие программ с операционной системой;
- понимать на базовом уровне как устроен механизм “песочниц” и “контейнеров”;
- разрабатывать собственные файловые системы;
- более тонко использовать возможности параллелизма;
- писать простейшие модули ядра Linux.

Разработка компиляторов

5–8 семестры (дисциплина по выбору).
3–4 зачётные единицы (в зависимости от отчётности).
зачёт или экзамен.

Аннотация

Данный курс представляет собой введение в область разработки компиляторов, языковых инструментов и языков программирования. Он предназначен для тех, кто интересуется внутренним устройством языков программирования, однако он также полезен для общего развития и более глубокого понимания операционных особенностей исполнения программ. На лекциях мы познакомимся с базовыми понятиями и походами области, такими, как семантики языков программирования, промежуточное представление программ, интерпретация, преобразования, оптимизация и т.д. На практике слушатели поэтапно реализуют полноценный компилятор простого языка императивного программирования в машинный код, начиная с компилятора арифметических выражений и простейших структур данных, заканчивая реализацией библиотеки поддержки времени исполнения с собственным менеджером памяти. Основным инструментом (реализации) в рамках курса является среда программирования `OCaml`.

Пререквизиты

- Теоретическая информатика 1: формальные языки.
- Низкоуровневое программирование на языке `C`.
- Функциональное программирование.
- Архитектура ЭВМ: основы архитектуры ЭВМ, основы программирования на ассемблере.

Где понадобится

- При реализации языков программирования, интерпретаторов, компиляторов, `DSL` и виртуальных машин

Содержание

- Устройство компилятора.
- Синтаксический анализ и парсер-генераторы.
- Основы семантики языков программирования, корректность компиляции.
- Стековая машина.
- Генерация кода и символьный интерпретатор.
- Представление и компиляция выражений, операторов, конструкций управления, процедур, функций, массивов, строк, символьных выражений, сопоставления с образцом и функций высших порядков.
- Программный стек, соглашения о вызовах, библиотека поддержки времени исполнения.
- Менеджеры памяти, сборка мусора.

Вы научитесь

- формально рассуждать о языках программирования;
- реализовывать, интерпретировать и компилировать основные структуры и конструкции языков программирования.

Типы в языках программирования

5–8 семестры (дисциплина по выбору).
3–4 зачётные единицы (в зависимости от отчётности).
зачёт или экзамен.

Аннотация

Системы типов являются неотъемлемой частью большинства современных языков программирования. Они помогают обеспечивать безопасное взаимодействие различных частей программы, позволяя задавать формализованные спецификации и контролировать соответствие кода этим спецификациям на этапах компиляции и исполнения.

В рамках курса мы изучим широкий класс типовых формализмов и реализуем для них алгоритмы проверки и вывода типов. Для каждой из изучаемых систем мы обсудим минимальное число типовых аннотаций в программе, необходимых для разрешимости задачи вывода типов.

Также мы исследуем вопросы типобезопасности и обсудим концепции продвижения и сохранения типа. Мы изучим различия декларативных и алгоритмических описаний систем типов, преимущества правила вывода, управляемых синтаксисом, и удобство двустороннего вывода. Будут затронуты темы сильной и слабой нормализуемости и задача обитаемости типов.

Мы обсудим связь систем типов и логических систем как в теоретическом аспекте (изоморфизм Карри-Говарда), так и в прикладном (движки систем проверки доказательств и верификации програмного обеспечения).

Курс сопровождается десятью лабораторными работами, оформленными в виде модулей на платформе Stepik.

Пререквизиты

- Дискретная математика 1: доказательства по индукции.
- Функциональное программирование: лямбда-исчисление, язык Haskell.
- Математическая логика: исчисления высказываний и предикатов, интуиционизм.

Где понадобится

- В курсах по семантике языков программирования и верификации программного обеспечения.
- При реализации инструментов, анализирующих и модифицирующих код.

Содержание

- Простая система типов (Simple Type Theory) и ее расширения: типы произведений и сумм, сопоставление с образцом, тип записей и вариантные типы.
- Сильная и слабая нормализация. Тип оператора неподвижной точки.
- Проблема населенности типов и алгоритм Бен-Йелса перечисления обитателей.
- Типы-пересечения. Теоремы об эквивалентности нормализуемости и типизируемости.
- Подтипы, отношение вложения для типов. Декларативные и алгоритмические версии отношения подтипизации.
- Субструктурные системы типов: линейность, афинность, релевантность.
- Полиморфные типы. System F, ранги полиморфизма. Кодирование стандартных типов в System F.
- Экзистенциальные типы. Понятие пакета (модуля). Абстрактные типы данных.
- Система с операторами над типами и полная System F_ω.
- Зависимые типы. Типы зависимых произведения и суммы.
- Лямбда-куб и чистые системы типов (PTS).
- Связь систем типов и логических систем.

Вы научитесь

- формализовывать отношение типизации для вычислительной системы в виде набора правил;
- реализовывать алгоритмы вывода, проверки и населенности типа для различных систем типов;
- анализировать системы типов на предмет обеспечения типобезопасности.

Трёхмерное компьютерное зрение

5–8 семестры (дисциплина по выбору).
3–4 зачётные единицы (в зависимости от отчётности).
зачёт или экзамен.

Аннотация

Компьютерное зрение — дисциплина, которая объединяет в себе различные задачи извлечения информации из изображений и видео, возникающие во множестве областей: от промышленности до кино. Под трехмерным компьютерным зрением понимают ту подобласть этой дисциплины, которая связана с восстановлением 3D-структуры снятого на камеру пространства.

В курсе рассматриваются идеи и классические алгоритмы компьютерного зрения. Порядок изложения построен вокруг емкого примера — проблемы извлечения позиций камеры и 3D-структуры сцены из видео. Практическая часть курса включает в себя поэтапную реализацию приложения, решающего эту задачу.

Пререквизиты

- Алгебра 1 и 2: линейная алгебра.
- Математический анализ 1 и 2: формула Тейлора, задачи оптимизации.
- Теория вероятностей.
- Алгоритмы и структуры данных 1.
- Основы программирования.
- Основы Linux.
- Уверенное владение языком программирования Python на базовом уровне (основные синтаксические структуры, ООП, модули, использование библиотек, умение читать документацию).

Где понадобится

- При изучении других курсов по компьютерному зрению.
- В работе в области компьютерного зрения и смежных сферах.

Содержание

- Азы обработки изображений, линейные фильтры и свертки.
- Ключевые точки: детекторы, дескрипторы, сопоставление, оптический поток.
- Камера: устройство, внутренние и внешние параметры.
- Вычисление облака точек и движения камеры.
- PnP и Bundle Adjustment как задачи оптимизации, алгоритм Левенберга — Марквардта.
- Робастное оценивание: RANSAC, M-оценки.
- Современные пайплайны SfM на примере COLMAP.

Вы научитесь

- азам обработки изображений;
- базовым подходам и идеям классического компьютерного зрения;
- на практике применять теоретические знания, полученные на математических курсах;
- понимать, как работает камера, какая информация теряется при съемке и как эту информацию можно восстановить;
- решать задачи компьютерного зрения на Python с помощью библиотек OpenCV, NumPy и пр.

Численные методы

5–8 семестры (дисциплина по выбору).
3–4 зачётные единицы (в зависимости от отчётности).
зачёт или экзамен.

Аннотация

Численные методы — это область науки, которая располагается между чистой математикой и высокопроизводительными вычислениями. Ученый и инженер, работающие в этой области, полагаются в равной степени на свои математические способности и умение программировать. Архитектура приложения отходит на второй план, а на первом плане оказывается умение написать алгоритм в рамках заданной математической модели и корректно запрограммировать его.

Задачи, решаемые с помощью численных методов, так же как и с помощью высокопроизводительных вычислений, часто являются специфичными для конкретной области науки: математики, физики, биологии, географии, геометрии и т.д. Каждая область может содержать математический аппарат и алгоритмы, отличающиеся от других областей и требующие длительного изучения. Это создает сложность в изучении численных методов: если рассматривать только базовые методы, применяющиеся почти во всех областях науки, то создастся впечатление, что кроме решения уравнений и матричных операций в численных методах ничего нет; если же изучать только одну конкретную область (например, численные схемы), то создастся впечатление, что численные схемы и есть численные методы, хотя это только одна подобласть.

В рамках данного курса была сделана попытка соблюсти баланс; курс состоит из трех больших частей:

- базовые методы,
- методы из конкретных областей,
- реальные задачи.

В первой части изучаются наиболее распространенные методы, которые одинаковы для всех областей. Во второй части изучаются методы из нескольких знакомых автору областей. В последней части изучаются задачи, для решения которых используются комбинации методов из предыдущих частей. Большинство заданий основаны на реальных задачах, с которыми автор курса сталкивался в профессиональной деятельности.

Пререквизиты

- Программирование на C++.
- Основы Linux.

Где понадобится

- Для разработки программ, моделирующих физические процессы (течение жидкости, волнообразование, горение, качка судна).
- Для разработки программ, решающих геометрические задачи (моделирование движения и столкновений трёхмерных объектов, трассировка лучей).
- Для разработки программ, решающих математические задачи (интерполяция, интегрирование, дифференцирование, оптимизация).

Содержание

- Понятие погрешности.
- Локальная и глобальная оптимизация.
- Интерполяция на регулярной и нерегулярной сетке.
- Численное интегрирование, дифференцирование и обыкновенные дифференциальные уравнения.
- Дифференциальные уравнения в частных производных на регулярных и нерегулярных сетках.
- Линейные алгебраические уравнения.
- Библиотеки для численных методов.
- Элементы вычислительной геометрии.

Вы научитесь

- корректно переносить математические модели в код программ;
- использовать физическую интуицию для решения задач;
- контролировать погрешность вычислений.

Язык программирования Rust

5–8 семестры (дисциплина по выбору).
3–4 зачётные единицы (в зависимости от отчётности).
зачёт или экзамен.

Аннотация

Rust — современный язык, ориентированный на безопасность памяти, производительность и продуктивность разработчика. Он успешно занимает нишу в разработке высокопроизводительных и надежных серверных приложений, а также находит применение в самых разнообразных прикладных задачах, таких как низкоуровневые драйверы и прошивки микроконтроллеров, новейшие графические API, распределенные базы данных и даже браузерные вычисления.

Теоретическая часть курса затрагивает все концепции, необходимые для профессиональной разработки ПО на Rust: владение и заимствование, модульная система, элементы функционального и объектно-ориентированного программирования, полиморфизм, асинхронность и многопоточность, метапрограммирование, Unsafe Rust и многое другое.

Практическая часть состоит из нескольких мини-проектов различной направленности. Завершив данный курс, студенты получат опыт современной разработки ПО в самых разнообразных областях: от консольных утилит до веб-серверов.

Пререквизиты

- Программирование на языке C или базовое знание любого другого языка программирования.
- Основы Linux.

Где понадобится

- В работе разработчика на Rust.
- Для более эффективной разработки на других языках программирования.

Содержание

- Владение и заимствование.
- Параметрический полиморфизм, трейты и времена жизни.
- Пакеты, крейты и модули.
- Стандартная библиотека.
- Unsafe Rust и FFI.
- Многопоточность и async/await.

Вы научитесь

- читать и писать код на Rust;
- разрабатывать библиотеки и приложения;
- эффективно сочетать элементы функционального и объектно-ориентированного программирования.

Ответы на частые вопросы

Как мы учим программированию

Подготовка программистов на университетском уровне — это, разумеется, не только программирование, но в её основе лежит именно обучение программированию. Специфика программы «Современное программирование» на факультете МКН СПбГУ состоит в том, что к нам в основном приходят победители олимпиад по информатике, а значит они уже умеют программировать, причём на очень хорошем уровне. Правда, в этом уровне скрыты и некоторые проблемы, решать которые мы начинаем в первой же программистской дисциплине «Основы программирования», который читается два первых месяца обучения, в сентябре и октябре первого курса.

Целью спортивного программиста является написание кода, который быстро находит ответ на чётко специфицированную задачу. Как только контрольные тесты пройдены, об этом коде можно забыть. Мы же готовим программистов для промышленности, а их код живёт годами и десятилетиями, и такие программы следует писать совсем иначе. Заботы о быстродействии промышленного кода обычно начинаются только тогда, когда программа уже корректно работает (она протестирована!), да и возникают они не всегда, а только если это быстродействие оказывается необходимым для решения бизнес-задач и если эта необходимость явно подтверждена результатами профилирования. А вот о простоте и понятности кода следует думать при его написании практически всегда, ведь этот код будут читать и поддерживать множество разработчиков на протяжении долгого времени жизни проекта.

Курс «Основы программирования» посвящён тем принципам, которым необходимо следовать при написании кода промышленного качества. Постановки задач имеют проектный характер, то есть понятно, что нужно получить в итоге, но не указано, как к этой цели придти. В этом курсе практически нет строго специфицированных задач, как нет их и в реальной практике разработки программного обеспечения. Зато в нём есть строгие критерии проверки качества. Весь код студентов преподаватели просматривают глазами и пишут по нему обзор недостатков (code review). Тут-то и выясняется, что программы требуют тщательного структурирования, продумывания удобных интерфейсов (даже консольных!), комментирования и документирования, устойчивости к ошибкам ввода и многого другого. Разумеется, программы должны выдавать правильные ответы, только заботиться об этом следует программисту, а не тестирующей системе или преподавателю! QA-отделы появятся в жизни будущего профессионального программиста несколько позже. На этом курсе студенты пишут программы на языке Kotlin. Это удобно, поскольку большинство поступающих с ним незнакомы, а значит они начинают в примерно равных условиях. Разумеется, на этом курсе никто не учит студентов писать циклы, но вот как показывает практика, написанию хороших функций учить уже приходится.

В середине первого семестра обучение программированию как бы раздваивается. С одной стороны курс по Kotlin переходит в объектно-ориентированное программирование и проектирование, параллельно с которыми изучаются удобные языковые инструменты и библиотеки. С другой стороны начинается изучение языка C. Параллельное изучение низкоуровневого и высокоуровневого языков позволяет студентам осознать широту доступных технологий. Весь второй семестр параллельно изучаются Kotlin (уже как инструмент промышленного программирования) и C++. В этот момент на Kotlin пишутся уже довольно большие программы.

В третьем семестре в нашем учебном плане стоит курс функционального программирования, читающийся на основе языка Haskell. Важно понимать, что в настоящее время не существует главенствующего стиля программирования, практически все современные языки программирования являются мультипарадигменными, они позволяют сочетать разные стили в рамках одной программы. Изучение чисто функционального языка Haskell позволяет здесь лучше понять, в каких ситуациях функциональное программирование оказывается более подходящим.

К концу третьего семестра освоение базовых инструментов программирования заканчивается. Три изученных языка, Kotlin, C++ и Haskell, формируют базу, на основе которой студенты способны самостоятельно выбирать наиболее подходящий для решения поставленной задачи инструмент, профессионально и качественно его применить. В этот момент мы начинаем проектную работу, где все эти знания оказываются необходимыми. Разумеется, и на старших курсах изучение программирования продолжается. В частности, наш учебный план предусматривает спецкурсы по различным языкам программирования (Python, Scala, Rust и др.) и деталям их реализации (курсы по компиляторам и архитектуре JVM), а также по их применению в различных сферах (от веба и мобильной разработки до компьютерной графики и обработки естественных языков). В седьмом семестре студентам читается обязательный курс программной инженерии. В этом курсе студенты знакомятся и практикуют инженерные навыки в области в разработки программного обеспечения, учатся планировать, оценивать время, организовывать работу, используя профессиональные методы.

Как мы организуем проектную работу

Проектная работа — один из важнейших способов подготовки программистов. Мы запускаем её в четвёртом семестре (весной второго курса), когда студенты объединяются в команды по несколько человек, выбирают задачу и технологии решения, организуют свою работу и в течение семестра её выполняют. В этой работе командам помогают менторы из числа опытных разработчиков, они и оказывают помощь в организации работы, смотрят и оценивают код, помогают его улучшить.

Иногда приходится слышать, что такого же рода проектную работу можно запускать и раньше, уже на первом курсе или даже в школе. Мы идём другим путём — раннее начало при недостаточно глубоком освоении инструментария может приводить к появлению решений, некачественных настолько, что даже опытные менторы не могут справиться с их улучшением. Представьте себе программиста, который и язык программирования должен осваивать, и детали реализации продумывать, и планов по проекту придерживаться. Очень сложно! К тому же писать плохой код вредно, даже если он решает требуемые задачи. Программист должен отвечать за качество своей работы и не должен привыкать к написанию некачественного кода.

С пятого семестра проекты становятся внешними, студенты выполняют их по задачам и под руководством разработчиков JetBrains, Яндекса и других компаний. Внешние проекты могут быть как групповыми, так и индивидуальными. Проект может иметь как прикладной, так и научно-исследовательский характер. Выбор исполнителей проектов происходит на конкурсной основе с участием студентов других образовательных программ, но опыт показывает, что наши студенты благодаря своей базе успешно справляются с тестами и собеседованиями.

Проекты выполняются в течение семестра и заканчиваются защитой, в рамках которой нужно не просто продемонстрировать результат, но и рассказать о ходе работы, проанализировать проблемы и сделать выводы. Выполняемые в течение пяти семестров проекты наполняют портфолио студента, ему есть что предъявить при подаче заявлений на летние стажировки в компаниях, летние и зимние школы, работодателям по окончании программы.

Мы считаем, что три семестра базовой подготовки, один подготовительный семестр внутренней проектной работы и четыре семестра внешних проектов в сумме формируют идеального разработчика, полностью готового к профессиональному решению любых возникающих в разработке ПО задач. Каждый из этих элементов важен, отказ от любого из них ухудшает качество подготовки.

Как мы учим математике, информатике и компьютерным системам

Вокруг программирования в программе обучения есть много всего. Это области, знания из которых, возможно, не будут применяться большинством студентами напрямую в их последующей работе, но которые позволяют сформировать образовательный фундамент и широкий кругозор и обеспечить гибкость в выборе карьерного пути. Даже в таких дисциплинах программирование оказывается вспомогательным навыком, необходимым тем не менее для их прохождения.

В нашем учебном плане четыре семестра математического анализа и два семестра алгебры. Эти дисциплины закладывают фундамент для изучения обязательных курсов по теории вероятностей, математической статистике и машинному обучению. Отдельные их элементы оказываются полезными, например, при выборе предлагаемых нами спецкурсов по выпуклой оптимизации, численным методам, компьютерной графике, квантовым вычислениям, биоинформатике и продвинутым методам машинного обучения. Мы не знаем, какая доля студентов углубится в математически-ёмкие области в своей профессиональной деятельности, но мы считаем обязательным дать им такую возможность. Наши преподаватели даже в математических курсах дают задания на программирование, демонстрируя тем самым наличие непосредственных приложений.

Два семестра дискретной математики, три семестра алгоритмов и структур данных, два семестра теоретической информатики (формальные языки, сложность вычислений, вычислимость, теория информации) — вот ядро теоретической базы современного программирования. К ним примыкает курс математической логики в информатике, выходящий в приложения в области верификации программного обеспечения и разработки доказуемо корректного ПО. Можно ли работать программистом без владениями материалом этих дисциплин? Безусловно, да. Но можно ли создавать принципиально новые подходы, алгоритмы и системы? Вряд ли.

Ещё три обязательные дисциплины нашего учебного плана — архитектура компьютера, операционные системы и компьютерные сети — позволяют студентам разобраться с работой современных компьютерных систем и понять, почему высокоуровневые подходы работают так, а не иначе. От чего зависит быстродействие программ, как устроена работа с кэшем процессора, как реализуется многопоточность, откуда возникает замедление при работе с вводом-выводом, как именно прикладные программы взаимодействуют с оборудованием, как данные передаются по сети — всё это являет-

ся предметом изучения этих дисциплин. К этой же группе дисциплин можно отнести базы данных, сейчас сложно представить себе полезную программу, которая не работает с базой данных, хотя бы локальной и самой простой. Как только все эти системные дисциплины изучены, мы предлагаем студентам курс проектирования высоконагруженных приложений, именно там все эти знания оказываются востребованными.

Как мы реализуем свободу выбора траектории обучения

Чем старше становится студент, тем лучше он начинает понимать, чем ему хотелось бы заниматься в будущем, в какие области углубляться. Иногда оказывается, что ему хочется сместиться в смежные области. Программист вполне может захотеть перейти в анализ данных или даже математику. Четверть всей нагрузки студента за всё время обучения определяется им полностью самостоятельно в рамках предлагаемых всеми образовательными программами факультета дисциплин по выбору. Мы не фиксируем траектории заранее, чтобы не ограничивать свободу выбора. В конце концов никто лучше, чем сам студент, не знает, что ему хочется изучить. Любые комбинации дисциплин здесь уместны. Смысл бакалавриата, по нашему мнению, в том, чтобы избежать строгой специализации и попробовать как можно больше разных вариантов. Ведь иногда, чтобы найти интересующую тебя область, нужно перебрать десятки других!

Дисциплины по выбору начинаются в пятом семестре и предлагаются до конца обучения в бакалавриате. Половина нагрузки в этих семестрах — это дисциплины по выбору. Свой первый выбор студенты делают весной второго курса, в этот момент они определяются с теми дисциплинами, которые планируют послушать на третьем курсе. Весной третьего курса делается выбор на четвёртый курс. Списки потенциально доступных на следующий учебный год дисциплин формируются заранее. Мы информируем студентов о содержании дисциплин и их преподавателях, рассказываем о правилах и порядке выбора, и предлагаем сделать выбор самостоятельно. Но мы всегда готовы помочь с выбором и посоветовать сочетания и последовательности предметов.

Дополнительный источник свободы — это выбор проектов с пятого по восьмой семестр. При желании можно все проекты брать в одной области и даже у одного руководителя, а можно и варьировать их, расширяя тем самым свой кругозор и внешние связи. Свободный выбор дисциплин и смена проектов способствуют поиску «своей» темы, то есть того, что именно интересно в профессии. К тому же это формирует более сильное портфолио и широкую базу знаний, а значит даёт возможность попасть на любые стажировки. В результате к моменту окончания бакалавриата студент понимает, чем он хочет заниматься (или чем точно не хочет!) и куда ему дальше стоит развиваться.

Как мы относимся к работе во время учёбы

Плохо.

Почему студенты хотят работать во время учёбы? Есть два стандартных ответа на этот вопрос. Первый ответ — необходимо заранее получить опыт работы, чтобы успешно конкурировать на рынке труда после завершения обучения. Но ведь для решения этой задачи есть и другие способы! Во-первых, наши семестровые проекты — это тот же опыт. Студент в течение четырёх месяцев выполняет проект по заданию и под присмотром опытных разработчиков. Если зарекомендовать себя на таком проекте хорошо, то можно считать, что место работы уже в кармане — любой компании куда проще и приятнее взять на работу человека, про которого известно, как он справляется с работой, чем брать кота в мешке со свободного рынка. Во-вторых, мы рекомендуем подаваться на летние стажировки для студентов в крупных компаниях. Стажировку лучше всего проходить после третьего курса, освободив время после первого и второго на летние школы. Сначала учёба, потом работа. Кто выигрышнее смотрится на рынке труда? Тот, кто два года проработал в малоизвестной компании, или тот, кто за время обучения прошёл стажировки в JetBrains, Яндексе или Google и может похвастаться хорошими рекомендациями? Впрочем, вряд ли эти двое даже будут конкурировать между собой, ведь запросы и возможности второго окажутся гораздо выше.

Второй стандартный ответ на вопрос о необходимости работы во время учёбы — потребности в средствах к существованию. Разумеется, это важная проблема, которую мы решаем повышенными стипендиями и хорошим общежитием. Если вы тратите всё своё время на учёбу и демонстрируете отличные результаты, то ваша стипендия позволит вам нормально жить. Если вы параллельно работаете и что-то зарабатываете, то неминуемо тратите меньше времени на учёбу. Вы тем самым выигрываете финансово на коротком отрезке, но проигрываете на длинном, теряя будущие возможности. Времени, которое можно будет полностью тратить на обучение, в вашей жизни больше не будет. Этот ресурс невосполним, а вот времени на работу впоследствии будет вдоволь. Наша программа обучения очень насыщенная. Даже если вам хватает времени на минимальное её осво-

ение, то, возможно, стоит попробовать максимум и выполнять все дополнительные задания или брать факультативные курсы, эта стратегия гарантированно себя окупит. Кстати, средние баллы за время обучения легко могут сыграть при поступлении в магистратуру (и закрыть некоторые двери). К сожалению, бывают ситуации, когда без работы во время учёбы никак не получается. Они часто сопровождаются проблемами в семье. Если вы всё-таки решаете работать, пожалуйста, убедитесь, что это ваша ситуация, но не пытайтесь играть только лишь в финансовую оптимизацию на коротком сроке. Это заведомо проигрышная стратегия. Нам кажется, что мы создаём достаточные условия для того, чтобы нашим студентам не приходилось работать во время учёбы. Разумеется, мы никак не препятствуем, если вы всё-таки решаете работать, но и не идём на встречу, если работа вдруг начинает мешать учёбе. Кстати, летние стажировки обычно неплохо оплачиваются, поэтому можно сделать запас на учебный год.

Как мы взаимодействуем со студентами

Мы как организаторы программы никогда не оставляем студентов наедине с вопросами и проблемами, учебными и не только. У нас есть регулярные встречи со студентами разных курсов «без темы» и тематические встречи (например, по представлению спецкурсов, психологической поддержке, продолжению учёбы за рубежом, стажировкам). Есть анонимные опросы о качестве преподаваемых дисциплин, они проводятся дважды за семестр, в середине и в конце. Результаты опросов всегда тщательно обрабатываются, по итогу принимаются решения. Бывают встречи с отдельными студентами и малыми группами. И вообще мы всегда на связи и охотно отвечаем на любые вопросы в почте, факультетском слаке, телеграме — где угодно. Обратная связь от студентов — один из важнейших инструментов развития образовательной программы. Наши преподаватели также всегда готовы помочь. У студентов есть наставники со старших курсов, они могут подсказать, к кому стоит обращаться при возникновении проблем. На факультете активно действует студенческий совет, крайне эффективный инструмент представления интересов студенчества. Чтобы создать атмосферу для общения между студентами разных курсов, мы организуем регулярные выезды на природу (и шашлыки!), иногда собираемся поиграть в настольные игры. В том же помогают и спецкурсы, которые одновременно посещают студенты разных курсов и всех образовательных программ факультета.