

# Программирование

**Лекторы:**

Григорий Французов

**Ассистент:**

Дарья Радзивил

**Язык:**

Русский

**Трудоемкость:**

6 з.е.

**Форма контроля:**

Дифф. зачет в 1 семестре

Экзамен во 2 семестре

**Образовательная программа:**

Теоретическая и экспериментальная физика

1, 2 семестры

Беспроводные технологии

1,2 семестры

Лекции (ак.час)*	Практические занятия (ак.час)	Лабораторные занятия (ак.час)
64	64	
*1 академический час = 45 минутам		

В курсе программирования мы познакомимся с современным языком программирования Python, научимся решать как алгоритмические задачи, так и задачи физического моделирования, линейной алгебры и математического анализа. Изучим основные модули (библиотеки) языка, троенные структуры данных.

# Содержание курса

## 1 семестр

### Программирование

#### Структура курса

Разделы	Лекции (ак.ч.)	Лабы (ак.ч.)
<b>1. Вводная: История и становление компьютеров</b>	2	2
<ol style="list-style-type: none"><li>1. Правила курса, рекомендуемые материалы.</li><li>2. Давид Гильберт, унификация математики. Курт Гедель, теорема о неполноте.</li><li>3. Машина Тьюринга, Лямбда Исчисление.</li><li>4. Декларативные (Функциональные) и Императивные языки.</li><li>5. Как построить исполнительное устройство, Архитектуры компьютеров: "код это данные" архитектура Фон Неймана "код отдельно от данных" Гарвардская архитектура</li><li>6. История компьютеров.</li><li>7. Устройство и физические ограничения современных компьютеров.</li></ol>		
<b>2. Данные и программирование: Переменные</b>	2	2
<ol style="list-style-type: none"><li>1. Программирование - преобразование данных из одной форму в другую.</li><li>2. Откуда они берутся, как и куда их можно передать, как их хранить.</li><li>3. Язык программирования Python, почему именно он.</li><li>4. Переменные и типы: числа, строки, коллекции</li><li>5. Хранение простых переменных в памяти.</li></ol>		
<b>3. Данные и программирование: Циклы и ветвления</b>	2	2
<ol style="list-style-type: none"><li>1. Булева алгебра</li><li>2. Ветвления в Python, нелинейные программы, оператор If</li><li>3. Сложные и комбинированные условия</li><li>4. Циклы (For и While)</li><li>5. Вычислительная сложность и циклы</li></ol>		
<b>4. Данные и программирование: Функции и рекурсия</b>	2	2
<ol style="list-style-type: none"><li>1. Что такое функции (абстракции), разбиение на функции</li><li>2. Аргументы функций</li><li>3. Возврат значений</li><li>4. Вызов функции из функции</li><li>5. Рекурсию</li><li>6. Передача функции в функцию (коллбэк, first class citizen) map, reduce</li><li>7. Стек вызовов, хранение функций в памяти</li></ol>		
<b>5. Данные и программирование: Словари и множества</b>	2	2
<ol style="list-style-type: none"><li>1. Словари, хранение в памяти</li><li>2. Расширение словаря</li><li>3. Перебор элементов словаря</li><li>4. Множества</li><li>5. Операции с множествами</li></ol>		
<b>6. Данные и программирование: Структурирование кода. Библиотеки и модули</b>	2	2
<ol style="list-style-type: none"><li>1. Пространство имен, модуль, пакет</li><li>2. Менеджер пакетов Pip, репозиторий пакетов</li><li>3. Управление зависимостями, циклические зависимости</li><li>4. Sandboxing, Dll hell</li></ol>		
<b>7. Процесс разработки ПО. Введение</b>	2	2

<ol style="list-style-type: none"> <li>1. Общее представление рабочем процессе</li> <li>2. Роли в команде разработки, варианты карьерного пути в ИТ</li> <li>3. Управление и планирование в командах</li> <li>4. Инструменты командной работы: <ul style="list-style-type: none"> <li>- таск трекаеры</li> <li>- система контроля версий,</li> <li>- CI/CD</li> <li>- мессенджеры и система коммуникаций</li> </ul> </li> <li>5. Вопросы безопасности</li> <li>6. Жизненный цикл ПО</li> </ol>		
<b>8. Процесс разработки ПО. Аналитика и постановка задач</b>	2	2
<ol style="list-style-type: none"> <li>1. Задача продуктовой аналитики</li> <li>2. Продуктовый дизайн</li> <li>3. Как построить диалог с заказчиком</li> <li>4. Как проверять идеи и гипотезы</li> <li>5. Customer Development</li> </ol>		
<b>9. Процесс разработки ПО. Профессия программиста и тестировщика</b>	2	2
<ol style="list-style-type: none"> <li>1. Уровни компетенций программиста</li> <li>2. Не программистские задачи (код ревью, менторинг)</li> <li>3. Отладка и дебаг (устранение неисправностей)</li> <li>4. Зачем нужно тестирование</li> <li>5. Виды тестирования, Пирамида тестирования</li> <li>6. Автоматизация тестирования, Unit-тесты</li> </ol>		
<b>10. Данные и программирование. Файлы</b>	2	2
<ol style="list-style-type: none"> <li>1. Работа с файлами в Python</li> <li>2. Подходы к хранению данных в файлах</li> <li>3. Сериализация и десериализация</li> <li>4. Распространенные форматы (CSV, JSON, XML)</li> <li>5. Библиотеки для работы с этими форматами</li> </ol>		
<b>11. Данные и программирование: Сети и сетевые протоколы</b>	2	2
<ol style="list-style-type: none"> <li>1. Компьютерные сети и их история</li> <li>2. Модели организации сетей. Модель OSI. TCP/IP, UDP</li> <li>3. Работа с сетевыми запросами в Python</li> <li>4. Интернет, DNS</li> <li>5. Сетевые протоколы, HTTP</li> <li>6. Работа с HTTP, библиотека requests</li> </ol>		
<b>12. Данные и программирование: Устройство современного веб приложения</b>	2	2
<ol style="list-style-type: none"> <li>1. Типы клиентов веб приложений</li> <li>2. Браузеры и их работа</li> <li>3. Подходы к проектированию веб приложений (MVC\MVVM)</li> <li>4. Разделение обработки и представления (бэкенд и фронтенд)</li> <li>5. Адаптивная верстка, mobile first</li> <li>6. Хостинг и облака</li> </ol>		
<b>13. Данные и программирование: API</b>	2	2
<ol style="list-style-type: none"> <li>1. Введение в паттерны проектирования веб приложений</li> <li>2. Подходы к проектированию API</li> <li>3. RPC, REST, GraphQL</li> <li>4. паттерн CQRS</li> <li>5. Веб API в Python. Flask/FastAPI</li> </ol>		
<b>14. Данные и программирование: Базы данных</b>	2	2

<ol style="list-style-type: none"> <li>1. Зачем нужны базы данных</li> <li>2. Типы баз данных</li> <li>3. Целостность данных ACID, транзакции</li> <li>4. CAP теорема</li> <li>5. Популярные базы данных и их особенности</li> </ol>		
--	--	--

## 2 семестр

### Программирование

#### Структура курса

Разделы	Лекции (ак.ч.)	Лабы (ак.ч.)
<b>1. Алгоритмы и структуры данных: Асимптотическая сложность простых алгоритмов</b>	2	2
<ol style="list-style-type: none"> <li>1. Базовые алгоритмы сортировки</li> <li>2. Сложность операций над структурами данных</li> </ol>		
<b>2. Алгоритмы и структуры данных: Алгоритмы поиска</b>	2	2
<ol style="list-style-type: none"> <li>1. Двоичный поиск</li> <li>2. Двоичная куча</li> <li>3. Деревья поиска</li> </ol>		
<b>3. Алгоритмы и структуры данных: Алгоритмы на графах</b>	2	2
<ol style="list-style-type: none"> <li>1. NP трудные задачи</li> <li>2. Методы оптимизации: <ul style="list-style-type: none"> <li>• монте-карло</li> <li>• градиентный спуск</li> <li>• имитация отжига</li> </ul> </li> </ol>		
<b>4. Объектно-ориентированное программирование: Введение и основные понятия</b>	2	2
<ol style="list-style-type: none"> <li>1. Что такое и зачем нужно ООП</li> <li>2. История ООП</li> <li>3. Принципы ООП</li> <li>4. SOLID</li> </ol>		
<b>5. Объектно-ориентированное программирование: Принципы ООП и особенности Python</b>	2	2
<ol style="list-style-type: none"> <li>1. Особенности и проявление принципов ООП в Python: <ul style="list-style-type: none"> <li>- объекты, классы, ЖЦ объекта</li> <li>- перегрузка методов</li> <li>- множественное наследование</li> </ul> </li> <li>2. ORM в Python</li> </ol>		
<b>6. Объектно-ориентированное программирование: Другие языки и подходы</b>	2	2
<ol style="list-style-type: none"> <li>1. Реализация в Smalltalk</li> <li>2. Реализация в JS</li> </ol>		
<b>7. Функциональное программирование</b>	2	2
<ol style="list-style-type: none"> <li>1. Главное отличие от ООП (ФП лучше/удобнее выражает процесс)</li> <li>2. Компиляторы,</li> <li>3. Автоматизированные доказательства</li> <li>4. (ФП как полигон для фиц других языков)</li> </ol>		
<b>8. Современные операционные системы: Взгляд программиста</b>	2	2

<ol style="list-style-type: none"> <li>1. Процесс загрузки</li> <li>2. Прерывания</li> <li>3. Системные вызовы</li> <li>4. Вытесняющая многозадачность</li> <li>5. Виртуальная память</li> <li>6. Кольца защиты</li> </ol>		
<b>9. Компиляторы и интерпретаторы</b>	2	2
<ol style="list-style-type: none"> <li>1. Подходы к архитектуре компиляторов</li> <li>2. Подходы к архитектуре интерпретаторов</li> </ol>		
<b>10. Concurrency и параллелизм</b>	2	2
<ol style="list-style-type: none"> <li>1. Что такое параллелизм</li> <li>2. Отличие concurrency от параллелизма</li> </ol>		
<b>11. Многопоточность и асинхронность: Теоретические основы</b>	2	2
<ol style="list-style-type: none"> <li>1. Отличие многопоточности и асинхронности</li> <li>2. Паттерны работы с многопоточными приложениями: <ul style="list-style-type: none"> <li>- семафор</li> <li>- mutex</li> <li>- замыкание</li> </ul> </li> </ol>		
<b>12. Многопоточность и асинхронность: Python</b>	2	2
<ol style="list-style-type: none"> <li>1. Библиотека Asyncio</li> <li>2. Паттерн Async/await</li> </ol>		
<b>13. Собеседования в IT компаниях</b>	2	2
<ol style="list-style-type: none"> <li>1. Выбор компании</li> <li>2. Процесс собеседования</li> <li>3. Подготовка к собеседованиям</li> <li>4. Переговоры с потенциальным работодателем</li> </ol>		
<b>14. Обработка данных в Python</b>	2	2
<ol style="list-style-type: none"> <li>1. Библиотека Pandas</li> <li>2. Библиотека NumPy</li> </ol>		
<b>15. Нейросети, прикладной взгляд</b>	2	2
<ol style="list-style-type: none"> <li>1. Введение в нейросети, какие задачи решают</li> <li>2. Базовые понятия</li> <li>3. Примеры и библиотеки</li> </ol>		

### Рекомендуемые ресурсы

1. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. — СПб.: Питер, 2015. — 1120 с.: ил. — (Серия «Классика computer science»). ISBN 978-5-496-01395-6
2. Лутц М. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО «Диалектика», 2019. — 832 с. : ил. — Парад, тит. англ. ISBN 978-5-907144-52-1 (рус., том 1)
3. Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. Алгоритмы: построение и анализ, 3-е издание = Introduction to Algorithms, Third Edition. — М.: «Вильямс», 2013. — 1328 с. — ISBN 978-5-8459-1794-2.

### Политика оценивания

Оценочные средства дисциплины: домашние задания, коллоквиум, дифф. зачет (1 семестр), экзамен (2 семестр).

В целом за семестр можно набрать 100 баллов:

- домашние задания - совокупно 60 баллов
- коллоквиум - максимум 10 баллов
- дифф. зачет/экзамен - максимум 30 баллов

Домашние задания оцениваются:

- Д/з № 1 - максимум 12 баллов,
- Д/з № 2 - максимум 15 баллов
- Д/з № 3 - максимум 15 баллов
- Д/з № 4 - максимум 18 баллов

Коллоквиум состоит из двух частей:

- 1) Теоретическо-практическая часть (тестовая форма).
- 2) Практическая часть, которая сдаётся с помощью средств git и GitHub.

Дифф. зачет/экзамен состоит из трех частей:

- 1) Теоретические вопросы. Несколько вопросов по курсу в виде диалога со студентом. Обязательная и основная часть проверки знаний.
- 2) Вопросы по домашним заданиям. Часть, которая отвечает за проверку осознанного и самостоятельного выполнения практических заданий во время семестра. Для некоторых студентов эта часть необязательна - при наличии хорошего отзыва от преподавателя практики.
- 3) Часть для тех, кто плохо ответил на предыдущие вопросы, но хочет хорошую отметку или повысить её. Включает в себя решение задач на языке Python и дополнительные вопросы.

Общая итоговая оценка формируется исходя из количества баллов: от 90 до 100 - «отлично», от 74 до 90 - «хорошо», от 60 до 74 - «удовлетворительно».