|  |  |  |
| --- | --- | --- |
| Gasy_1 | Министерство образования и науки Российской Федерации  Федеральное государственное бюджетное образовательное учреждение  высшего образования  САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ  АРХИТЕКТУРНО-СТРОИТЕЛЬНЫЙ УНИВЕРСИТЕТ |  |

Кафедра информационных технологий

**План проекта дополнительной дисциплины**

«Современные технологии и алгоритмы: путь к олимпиадным победам»

|  |
| --- |
|  |
|  | Подготовил:  студент группы ИСТб-3  Ферштадт К.Д.  студент группы ИСТб-3  Корнюшин А.А.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Санкт-Петербург

2025 г.

**Оглавление**

[**Цель дисциплины 4**](#_Toc187245147)

[**Задачи 4**](#_Toc187245148)

[**1. Обучение базовым и продвинутым концепциям программирования 4**](#_Toc187245149)

[**2. Развитие алгоритмического мышления 4**](#_Toc187245150)

[**3. Подготовка к олимпиадам разного уровня (вузовские, международные) 5**](#_Toc187245151)

[**4. Освоение современных инструментов программирования (библиотеки, фреймворки) 6**](#_Toc187245152)

[**Структура курса 6**](#_Toc187245153)

[Модуль 0. Введение в языки программирования и базовые фреймворки 6](#_Toc187245154)

[Модуль 1. Базовые алгоритмы и структуры данных 7](#_Toc187245155)

[Модуль 2. Продвинутые алгоритмы 7](#_Toc187245156)

[Модуль 3. Основы работы с базами данных 7](#_Toc187245157)

[Модуль 4. Вёрстка и веб-дизайн 8](#_Toc187245158)

[Модуль 5. Практика и соревнования 9](#_Toc187245159)

[Модуль 6. Современные технологии 9](#_Toc187245160)

[**Различные стратегии развития студентов 10**](#_Toc187245161)

[Roadmap 1: Подготовка к олимпиадному программированию 10](#_Toc187245162)

[Roadmap 2: Полный цикл разработки веб-приложений 10](#_Toc187245163)

[Roadmap 3: Современные технологии и ML 10](#_Toc187245164)

[Roadmap 4: Комплексная подготовка с акцентом на C++ 11](#_Toc187245165)

[**Методы преподавания 11**](#_Toc187245166)

[1. Теоретические лекции (20%) 11](#_Toc187245167)

[2. Практические занятия с решением задач (50%) 12](#_Toc187245168)

[3. Мини-олимпиады и проекты (30%) 12](#_Toc187245169)

[**Элементы геймификации 13**](#_Toc187245170)

[**1. Система наград за успехи: бейджи, сертификаты 13**](#_Toc187245171)

[**2. Лидерборд для студентов, демонстрирующий успехи в заданиях 14**](#_Toc187245172)

[**3. Продвинутые элементы геймификации 15**](#_Toc187245173)

[**4. Пример геймифицированной системы курса 15**](#_Toc187245174)

[**Оценка знаний 16**](#_Toc187245175)

[**1. Тестирование по теории 16**](#_Toc187245176)

[**2. Решение практических задач на время 17**](#_Toc187245177)

[**3. Итоговый проект 17**](#_Toc187245178)

[**4. Участие в реальных олимпиадах 18**](#_Toc187245179)

[**Поддержка студентов 19**](#_Toc187245180)

[**Социальные аспекты и вклад 19**](#_Toc187245181)

[**1. Участие в реальных проектах 19**](#_Toc187245182)

[**2. Разработка приложений для социальных нужд 20**](#_Toc187245183)

[**3. Итоговые преимущества социальной направленности курса 21**](#_Toc187245184)

[**Обратная связь и оценка курса 21**](#_Toc187245185)

[**1. Сбор предложений и отзывов студентов 21**](#_Toc187245186)

[**2. Оценка качества курса 22**](#_Toc187245187)

[**3. Постоянная доработка курса 23**](#_Toc187245188)

[**4. Стратегия вовлечения студентов в улучшение курса 23**](#_Toc187245189)

[**5. Заключительная обратная связь 23**](#_Toc187245190)

[**Итоговые навыки 23**](#_Toc187245191)

[**Учебные материалы 24**](#_Toc187245192)

[Книги по основам языков программирования: 24](#_Toc187245193)

[Книги по веб-разработке: 26](#_Toc187245194)

[Книги по Алгоритмам: 29](#_Toc187245195)

[Книги по базам данных: 31](#_Toc187245196)

[Онлайн-ресурсы: 32](#_Toc187245197)

**Цель дисциплины**

Повышение уровня знаний в программировании, освоение современных инструментов разработки, подготовка к олимпиадам по программированию и развитие навыков решения сложных задач.

# **Задачи**

### ****1. Обучение базовым и продвинутым концепциям программирования****

#### **1.1. Базовые концепции**

* **Что входит**:
  + Типы данных (целые, строки, массивы, списки).
  + Управляющие структуры (условные операторы, циклы).
  + Основы функций и модульного программирования.
  + Основы работы с файлами и вводом/выводом данных.
* **Этапы реализации**:
  + Объяснение теоретических основ с примерами.
  + Решение простых задач, таких как "Написать функцию для вычисления факториала".
  + Применение знаний на практике через мини-проекты, например, создание простого калькулятора.
* **Инструменты**:
  + IDE.
  + Платформы для тестирования кода: Repl.it, Codeforces.

#### **1.2. Продвинутые концепции**

* **Что входит**:
  + Объектно-ориентированное программирование (ООП): классы, наследование, полиморфизм.
  + Параллельное программирование: многопоточность, асинхронность.
  + Работа с API и взаимодействие с внешними системами.
  + Паттерны проектирования: Singleton, Factory, Observer.
* **Этапы реализации**:
  + Теория с примерами кода: "Как реализовать наследование на практике".
  + Углублённые задачи, например, "Реализовать очередь с приоритетами".
  + Проекты: создание веб-приложений или игр с использованием ООП.
* **Инструменты**:
  + Библиотеки и фреймворки: NumPy, Django, Flask, Qt.
  + Репозитории: GitHub, GitLab.

### ****2. Развитие алгоритмического мышления****

#### **2.1. Значение**

* Формирует способность решать задачи через последовательные, оптимальные действия.
* Учит анализировать проблему, искать подходящие алгоритмы и реализовывать их на практике.

#### **2.2. Основные темы**

* Базовые алгоритмы: сортировки, поиск, жадные алгоритмы.
* Структуры данных: массивы, стеки, очереди, графы, деревья.
* Рекурсия и динамическое программирование.
* Алгоритмы на графах: поиск в глубину (DFS), поиск в ширину (BFS), алгоритмы Дейкстры и Крускала.

#### **2.3. Методы обучения**

1. **Анализ задач**:
   * Пример задачи: "Найти минимальное расстояние между двумя точками на плоскости".
   * Обсуждение, какие алгоритмы подойдут, и выбор оптимального.
2. **Реализация алгоритмов**:
   * Студенты кодируют алгоритмы, начиная с простых (сортировка вставками) и переходя к сложным (динамическое программирование).
3. **Задачи на оптимизацию**:
   * Пример: "Найти наибольшую сумму подмассива в массиве".
4. **Визуализация алгоритмов**:
   * Использование инструментов, таких как VisuAlgo, для демонстрации работы алгоритмов.

### ****3. Подготовка к олимпиадам разного уровня (вузовские, международные)****

#### **3.1. Уровни подготовки**

* **Начальный**:
  + Решение базовых задач на платформе, такой как Codeforces (Div. 3).
  + Изучение шаблонов решений простых задач (сортировки, работа с массивами).
* **Средний**:
  + Работа с задачами на динамическое программирование, графы.
  + Участие в вузовских олимпиадах.
* **Высокий**:
  + Решение задач уровня ICPC, Codeforces (Div. 1).
  + Подготовка к международным олимпиадам (IOI, ACM-ICPC).

#### **3.2. Методы подготовки**

1. **Регулярные тренировки**:
   * Каждую неделю проводится 2–3 тренировки с задачами разного уровня.
   * Пример задачи: "Максимальный поток в транспортной сети".
2. **Участие в виртуальных олимпиадах**:
   * Использование платформ, таких как AtCoder или TopCoder, для моделирования реальных условий.
3. **Разбор сложных задач**:
   * Преподаватель объясняет задачи, которые студенты не смогли решить, с пошаговым разбором.
4. **Обсуждение стратегий**:
   * Как правильно распределить время, какие задачи решать первыми.

### ****4. Освоение современных инструментов программирования (библиотеки, фреймворки)****

#### **4.1. Библиотеки**

* **NumPy/Pandas**: Работа с массивами и данными.
  + Пример: "Написать программу для обработки и анализа финансовых данных".
* **Matplotlib/Seaborn**: Визуализация данных.
  + Пример: "Создать график распределения результатов студентов".

#### **4.2. Фреймворки**

* **Django/Flask**: Разработка веб-приложений.
  + Пример: "Создать систему управления библиотекой".
* **Qt**: Создание десктопных приложений.
  + Пример: "Создать приложение для учёта расходов".

#### **4.3. Системы управления версиями**

* **Git/GitHub**:
  + Пример: "Совместная работа над проектом через pull requests".

#### **4.4. Работа с API**

* Изучение REST API: запросы, обработка данных.
  + Пример: "Получить данные о погоде с OpenWeatherMap API".

**Структура курса**

(структура курса является общей, в процессе обучения необходимо индивидуально исключать лишний материал и координировать дальнейший вектор обучения)

## **Модуль 0. Введение в языки программирования и базовые фреймворки**

1. **Введение в C++ и настройка среды разработки (Visual Studio 2022, CLion)**
   * Объяснение ключевых особенностей языка C++: типизация, управление памятью, работа с указателями.
   * Пошаговое руководство по установке Visual Studio и CLion.
   * Настройка среды для работы с проектами, компиляция и отладка.
2. **Введение в JavaScript и настройка среды разработки (VS Code, Node.js)**
   * Краткий обзор языка JavaScript, его применения в веб-разработке.
   * Установка VS Code, установка и настройка Node.js.
   * Создание и запуск первого JavaScript-скрипта.
3. **Введение в Python и настройка среды разработки (PyCharm, VS Code)**
   * Основы Python: синтаксис, переменные, работа с данными.
   * Установка PyCharm и Jupyter Notebook.
   * Практическая работа: написание и выполнение первых программ.
4. **Введение в Java и настройка среды разработки (IntelliJ IDEA, Eclipse)**
   * Изучение основных концепций Java: объектно-ориентированное программирование, работа с классами.
   * Установка IntelliJ IDEA и Eclipse, настройка окружения.
   * Разработка тестового консольного приложения.

## **Модуль 1. Базовые алгоритмы и структуры данных**

1. **Введение в олимпиадное программирование**
   * Основные принципы и подходы к решению олимпиадных задач.
   * Разбор типичных ошибок и стратегий оптимизации.
2. **Сортировки, поиск, алгоритмы на строках**
   * Изучение алгоритмов сортировки (быстрая, слиянием, пузырьком).
   * Бинарный поиск и его применение.
   * Основные алгоритмы работы со строками (поиск подстроки, хэши).
3. **Основные структуры данных (стек, очередь, дерево, граф)**
   * Обзор и реализация базовых структур данных.
   * Примеры использования в реальных задачах.
4. **Реализация алгоритмов в рамках практических задач**
   * Решение задач на Codeforces, LeetCode и AtCoder.
   * Имитация олимпиадных условий для закрепления навыков.

## **Модуль 2. Продвинутые алгоритмы**

1. **Динамическое программирование**
   * Базовые принципы: разделяй и властвуй, хранение промежуточных результатов.
   * Примеры решения олимпиадных задач.
2. **Работа с памятью**
   * Управление памятью в C++ и Java.
   * Разбор утечек памяти и способов их предотвращения.
3. **Графовые алгоритмы (поиск кратчайшего пути, обход графа)**
   * Обходы в ширину и глубину.
   * Алгоритмы Дейкстры и Флойда-Уоршелла.
4. **Комбинаторика и теоремы чисел**
   * Основы комбинаторики: перестановки, сочетания.
   * Теоретические аспекты чисел: простые числа, делимость.
5. **Построение оптимальных решений для сложных задач**
   * Разработка эффективных решений для задач высокой сложности.

## **Модуль 3. Основы работы с базами данных**

1. **Введение в базы данных**
   * Что такое базы данных и зачем они нужны.
   * Типы баз данных: реляционные (SQL) и нереляционные (NoSQL).
   * Примеры популярных систем (MySQL, PostgreSQL, MongoDB).
2. **Настройка и работа с MySQL**
   * Установка и базовая настройка MySQL.
   * Создание баз данных, таблиц и ввод данных.
   * Основные SQL-запросы: SELECT, INSERT, UPDATE, DELETE.
   * Практическая работа: создание простой базы данных для веб-приложения.
3. **Работа с PostgreSQL**
   * Особенности PostgreSQL и его преимущества.
   * Установка и настройка.
   * Работа с типами данных и транзакциями.
   * Практическая работа: проектирование более сложной структуры базы данных.
4. **NoSQL базы данных (MongoDB)**
   * Принципы работы и отличие от реляционных баз данных.
   * Установка MongoDB и знакомство с коллекциями.
   * CRUD-операции в MongoDB.
   * Практическая работа: создание небольшой базы данных для хранения данных приложения.
5. **Интеграция баз данных в проекты**
   * Подключение баз данных к Python (SQLAlchemy) или JavaScript (Sequelize).
   * Создание REST API с базой данных (Django или Flask).
   * Практическая работа: создание REST API для работы с базой данных.

## **Модуль 4. Вёрстка и веб-дизайн**

1. **Введение в HTML и CSS**
   * Основы веб-структуры (HTML).
   * Стилизация страниц с помощью CSS.
2. **Практическое задание: создание тестового веб-сайта**
   * Разработка сайта с использованием HTML/CSS.
   * Адаптивная верстка под мобильные устройства.
3. **Введение в Angular, Vue.js, React (фронтенд)**
   * Сравнение фреймворков, их преимущества и недостатки.
   * Установка и настройка React для разработки.
4. **Практическое задание: разработка веб-приложения с использованием React**
   * Создание одностраничного приложения.
   * Интеграция с REST API.
5. **Введение в Django (фронтенд + бэкенд)**
   * Разработка серверной части веб-приложений.
   * Создание базовых API.
6. **Введение в Ruby on Rails, Laravel, Express.js (бэкенд)**
   * Обзор популярных фреймворков.
   * Практическая работа с выбором фреймворка.
7. **Введение в Flask (серверная часть)**
   * Разработка микросервисов на Python.
   * Подключение к базам данных.

## **Модуль 5. Практика и соревнования**

1. **Решение задач из олимпиад прошлых лет**
   * Разбор наиболее сложных задач.
   * Тренировка на задачи ACM и IOI.
2. **Имитация соревновательного программирования**
   * Организация тренировочных сессий.
   * Использование систем тестирования.
3. **Проведение мини-олимпиад с задачами разного уровня**
   * Создание и проведение внутренних соревнований.
   * Анализ решений и выбор оптимальных подходов.

## **Модуль 6. Современные технологии**

1. **Практика с Python, JavaScript, C++ или Java (на выбор)**
   * Работа над проектами с использованием выбранного языка.
   * Практическое применение знаний.
2. **Введение в машинное обучение (основы TensorFlow, PyTorch)**
   * Установка библиотек и подготовка данных.
   * Создание первых моделей машинного обучения.
3. **Введение в разработку веб-приложений (работа с REST API, фреймворки)**
   * Построение клиент-серверного взаимодействия.
   * Тестирование и деплой API.
4. **Введение в разработку графических приложений (OpenGL, Qt)**
   * Основы графики и визуализации.
   * Разработка простых графических интерфейсов.
5. **Проекты на основе современных технологий для закрепления навыков**
   * Завершающий проект с использованием изученных технологий.

# **Различные стратегии развития студентов**

## **Roadmap 1: Подготовка к олимпиадному программированию**

Цель: Углубленное изучение алгоритмов, структур данных и соревновательного программирования.

1. **Неделя 1-2**
   * Модуль 0.1: Введение в C++ и настройка среды разработки.
   * Модуль 1.1: Введение в олимпиадное программирование.
   * Модуль 1.2: Базовые сортировки, бинарный поиск.
2. **Неделя 3-4**
   * Модуль 1.3: Основные структуры данных (стек, очередь, дерево).
   * Модуль 1.4: Решение задач на Codeforces и LeetCode.
3. **Неделя 5-6**
   * Модуль 2.1: Динамическое программирование.
   * Модуль 2.3: Графовые алгоритмы (DFS, BFS).
   * Модуль 2.5: Построение оптимальных решений для сложных задач.
4. **Неделя 7-8**
   * Модуль 5.1: Решение задач из олимпиад прошлых лет.
   * Модуль 5.3: Участие в олимпиадах.

## **Roadmap 2: Полный цикл разработки веб-приложений**

Цель: создать веб-приложение с базой данных и REST API.

1. **Неделя 1-2**
   * Модуль 0.2: Введение в JavaScript и настройка среды (Node.js).
   * Модуль 0.3: Введение в Python и Django (фронтенд + бэкенд).
2. **Неделя 3-4**
   * Модуль 3.1: Введение в базы данных (SQL).
   * Модуль 3.2: Настройка и работа с MySQL.
3. **Неделя 5-6**
   * Модуль 4.3: Введение в React.
   * Модуль 4.4: Разработка веб-приложения с React + интеграция с REST API.
4. **Неделя 7-8**
   * Модуль 3.5: Интеграция баз данных в проект (SQLAlchemy).
   * Модуль 5.5: Завершающий проект: полноценное веб-приложение с REST API и базой данных.

## **Roadmap 3: Современные технологии и ML**

Цель: Разработка проекта на основе машинного обучения.

1. **Неделя 1-2**
   * Модуль 0.3: Введение в Python и настройка Jupyter Notebook.
   * Модуль 6.2: Введение в машинное обучение (TensorFlow, PyTorch).
2. **Неделя 3-4**
   * Модуль 6.3: Работа с REST API для проектов ML.
   * Модуль 3.4: Интеграция с NoSQL базами данных (MongoDB).
3. **Неделя 5-6**
   * Модуль 6.5: Завершающий проект: разработка приложения, использующего модель ML.
   * Пример: прогнозирование продаж или анализ данных.

## **Roadmap 4: Комплексная подготовка с акцентом на C++**

Цель: Углубленное изучение C++ для промышленного программирования.

1. **Неделя 1-2**
   * Модуль 0.1: Настройка Visual Studio, основы C++.
   * Модуль 1.3: Основные структуры данных (стек, очередь, дерево).
2. **Неделя 3-4**
   * Модуль 2.2: Работа с памятью в C++.
   * Модуль 2.4: Комбинаторика и теоремы чисел.
3. **Неделя 5-6**
   * Модуль 6.4: Введение в Qt.
   * Практическая разработка графического интерфейса.
4. **Неделя 7-8**
   * Модуль 6.5: Завершающий проект: графическое приложение с использованием C++ и Qt.

# **Методы преподавания**

## **1. Теоретические лекции (20%)**

#### **1.1. Роль теоретических лекций**

* Знакомят студентов с новыми концепциями, подходами и инструментами.
* Формируют базовую основу для практических занятий.
* Помогают понять принципы и логику работы различных алгоритмов, технологий и инструментов.

#### **1.2. Формат лекций**

1. **Классические лекции с презентациями**:
   * Использование визуальных материалов (слайды, схемы, графики).
   * Пример: Лекция о структурах данных с демонстрацией их работы на интерактивных диаграммах.
2. **Интерактивные лекции**:
   * Постановка вопросов студентам во время лекции для вовлечения.
   * Пример: Преподаватель объясняет алгоритм и просит студентов предложить оптимизации.
3. **Видео-лекции**:
   * Записанные лекции, которые студенты могут пересматривать в любое время.
   * Пример: Видео с объяснением основ SQL-запросов с примерами кода.

#### **1.4. Инструменты для лекций**

* PowerPoint/Keynote для создания презентаций.
* Онлайн-доски (Miro, Microsoft Whiteboard) для интерактивных объяснений.
* Видеозапись лекций с помощью OBS Studio или Camtasia.

#### **1.5. Способы оценки усвоения лекционного материала**

* Короткие тесты после лекции.
* Задания на самостоятельное изучение.
* Вопросы на проверку понимания на следующих занятиях.

## **2. Практические занятия с решением задач (50%)**

#### **2.1. Роль практических занятий**

* Развивают навыки применения теории для решения реальных задач.
* Формируют мышление, необходимое для оптимизации и анализа решений.
* Повышают уверенность студентов в их практических умениях.

#### **2.2. Формат практических занятий**

1. **Решение задач в классе**:
   * Студенты решают задачи под руководством преподавателя.
   * Пример: реализовать сортировку пузырьком и сравнить её производительность с сортировкой слиянием.
2. **Групповые работы**:
   * Студенты работают в командах, что помогает развивать навыки сотрудничества.
   * Пример: Разработка простой программы учёта задач (TODO-лист).
3. **Работа с проектами**:
   * Студенты работают над проектами, связанными с реальными задачами.
   * Пример: Создание базы данных для управления школьной библиотекой.
4. **Алгоритмические практикумы**:
   * Решение задач на LeetCode, Codeforces или HackerRank в реальном времени.
   * Пример: Задачи на динамическое программирование или жадные алгоритмы.

#### **2.3. Оценка практических занятий**

* Проверка точности и оптимальности решений.
* Время, затраченное на решение задачи.
* Качество кода (читаемость, стиль, комментарии).

## **3. Мини-олимпиады и проекты (30%)**

#### **3.1. Роль мини-олимпиад и проектов**

* Укрепляют навыки работы в стрессовых условиях.
* Позволяют студентам применить знания в реальных ситуациях.
* Мотивируют через дух соревнования и реальный результат (проекты).

#### **3.2. Мини-олимпиады**

1. **Формат проведения**:
   * Участвуют как индивидуальные участники, так и команды.
   * Пример: 3 часа на решение 5 задач разной сложности.
2. **Примеры задач**:
   * **Лёгкая задача**: найти все простые числа до 1000.
   * **Сложная задача**: реализовать алгоритм для нахождения минимального остовного дерева в графе.
3. **Оценка олимпиад**:
   * Баллы за каждый правильный ответ.
   * Дополнительные баллы за скорость и оптимальность.

#### **3.3. Проекты**

1. **Примеры проектов**:
   * Приложение для учёта расходов с базой данных.
   * Веб-приложение для планирования задач.
   * Игра на основе физических моделей.
2. **Этапы работы над проектом**:
   * Постановка задачи: Студенты формулируют проблему и цели.
   * Разработка прототипа: Простейшая версия программы.
   * Итеративная доработка: Добавление новых функций и оптимизация.
   * Финальная презентация: Демонстрация работы программы с объяснением решений.
3. **Критерии оценки проектов**:
   * **Функциональность**: Соответствие требованиям.
   * **Качество кода**: Чистота и читаемость.
   * **Документация**: Наличие инструкций и комментариев.
   * **Презентация**: Умение объяснить проект и продемонстрировать его работу.

# **Элементы геймификации**

**1. Система наград за успехи: бейджи, сертификаты**

**1.1. Бейджи**

* **Описание**: Виртуальные награды за достижение определённых целей.
* **Примеры категорий бейджей**:
  + **За мастерство**:
    - "Гуру динамического программирования" (за успешное решение 10 задач на динамическое программирование).
    - "Мастер баз данных" (за создание сложной структуры базы данных в PostgreSQL).
  + **За упорство**:
    - "Не сдаюсь!" (за успешное завершение всех заданий недели).
    - "Ночной кодер" (за сдачу заданий в ночное время).
  + **За скорость**:
    - "Ракета" (за самое быстрое решение задачи на уроке).
  + **Социальные бейджи**:
    - "Помощник команды" (за активную помощь другим студентам в чате).
    - "Признание сообщества" (за лучший ответ на форуме/вопросы).
* **Инструменты реализации**:
  + Визуализация бейджей в личном кабинете студента.
  + Возможность делиться бейджами в соцсетях (например, "Я стал Гуру Python!").
  + Использование платформ с API для выдачи бейджей, например, Badgr.

**1.2. Сертификаты**

* **Описание**: Официальные документы, подтверждающие успехи.
* **Примеры сертификатов**:
  + Сертификат об успешном завершении курса.
  + Сертификат "Топ-10 студентов" (для самых успешных участников).
  + Сертификат за участие в мини-олимпиадах.
* **Инструменты реализации**:
  + Автоматическая генерация сертификатов через платформы вроде Canva или Accredible.
  + QR-коды для проверки подлинности сертификатов.

**1.3. Награды для повышения мотивации**

* **Физические/материальные награды**:
  + Книги по программированию.
  + Бесплатный доступ к следующим модулям курса.
  + Призы для топовых студентов (флешки, футболки, сувениры).
* **Идея для вовлечённости**:
  + **"Тайные награды"**: студентам не сообщают заранее, за что даётся награда. Это стимулирует исследовать весь материал.

**2. Лидерборд для студентов, демонстрирующий успехи в заданиях**

**2.1. Что такое лидерборд**

* Это таблица, где отображаются успехи студентов, ранжированные по баллам или другим показателям.
* Лидерборд стимулирует соревновательный дух, мотивируя студентов улучшать свои результаты.

**2.2. Примеры показателей для лидерборда**

* Количество решённых задач.
* Время решения задач (задачи решены быстрее всех).
* Количество бейджей, заработанных студентом.
* Общий рейтинг на основе успешности выполнения модулей и активности.

**2.3. Форматы отображения**

* **Глобальный лидерборд**:
  + Показ успехов всех студентов курса.
  + Разделение на недели или модули.
* **Групповой лидерборд**:
  + Рейтинг внутри небольшой группы студентов.
  + Особенно полезен для курсов с большим количеством участников.
* **Персонализированный лидерборд**:
  + Отображение позиций ближайших конкурентов и своей текущей позиции.

**2.4. Инструменты для реализации**

* **Системы LMS** (например, Moodle, Google Classroom) с модулем геймификации.
* **Сервисы для создания лидербордов**:
  + Kahoot (встраиваемые тесты с рейтингом).
  + Rankade (для подсчёта и визуализации рейтингов).

**2.5. Уровни сложности**

* Создание **уровней**:
  + Например, 5 уровней: новичок → любитель → профессионал → эксперт → мастер.
  + Достижение следующего уровня требует определённого количества баллов или задач.

**3. Продвинутые элементы геймификации**

**3.1. Еженедельные/ежемесячные испытания**

* Создание задач или проектов с ограниченным временем выполнения.
* Примеры:
  + Разработка веб-приложения за 3 дня.
  + Решение 5 сложных алгоритмических задач за неделю.
* Победители получают дополнительные баллы, бейджи или сертификаты.

**3.2. Кооперативные задания**

* Командные проекты, где студенты делятся на группы.
* Взаимодействие стимулирует развитие софт-скиллов.
* Пример:
  + Совместная разработка REST API.
  + Решение задачи на оптимизацию, где каждый студент отвечает за определённый модуль.

**3.3. Прогресс-бар**

* Визуальный индикатор выполнения заданий курса.
* Например:
  + "Вы завершили 50% курса! Осталось совсем немного до следующего уровня!".

**4. Пример геймифицированной системы курса**

1. **Начало курса**:
   * Выдаётся первый бейдж: "Добро пожаловать в курс!".
   * Объявляется первая цель: завершить модуль 0.
2. **По ходу курса**:
   * За каждую завершённую тему/задание выдаются бейджи.
   * Лидерборд обновляется каждую неделю.
3. **По завершении курса**:
   * Топ-10 студентов получают сертификаты "Лучший студент".
   * Победители мини-олимпиад получают физические награды.

# **Оценка знаний**

### ****1. Тестирование по теории****

#### **1.1. Зачем нужны теоретические тесты?**

* Проверяют базовые знания, которые необходимы для выполнения практических заданий.
* Помогают выявить пробелы в понимании ключевых концепций.
* Способствуют запоминанию теории через повторение.

#### **1.2. Форматы тестов**

1. **Множественный выбор**:
   * Вопросы с несколькими вариантами ответа, из которых нужно выбрать один или несколько.
   * Пример: "Какой из следующих алгоритмов является жадным?"
2. **Краткий ответ**:
   * Вопросы, требующие короткого текстового ответа.
   * Пример: "Назовите асимптотическую сложность сортировки слиянием."
3. **Соответствие**:
   * Установление соответствия между понятиями, алгоритмами и их свойствами.
   * Пример: "Сопоставьте структуру данных с её примером применения."
4. **Верно/Неверно**:
   * Быстрые проверки знаний через утверждения.
   * Пример: "Динамическое программирование применимо только к графовым задачам. (Верно/Неверно)"

#### **1.3. Примеры тем для теоретических тестов**

* Основы языков программирования (C++, Python, JavaScript).
* Алгоритмы сортировки и поиска.
* Структуры данных (стек, очередь, дерево, граф).
* Основы работы с базами данных и SQL.
* Веб-разработка (HTML, CSS, REST API).

#### **1.4. Организация тестирования**

* Использование онлайн-платформ, таких как Google Forms, Kahoot, или специализированных систем вроде Moodle.
* Автоматическая проверка тестов для ускорения обратной связи.
* Ограничение по времени, чтобы оценить не только знания, но и быстроту мышления.

#### **1.5. Примерное содержание теста**

* **Вводный тест:** Проверка базовых знаний перед началом курса.
* **Промежуточные тесты:** Оценка усвоения материала после каждого модуля.
* **Финальный тест:** Итоговая проверка знаний по всему курсу.

### ****2. Решение практических задач на время****

#### **2.1. Зачем это нужно?**

* Проверяет способность применять теоретические знания на практике.
* Развивает умение быстро находить решения в стрессовых условиях.
* Моделирует реальные условия соревнований, таких как олимпиада.

#### **2.2. Форматы задач**

1. **Кодинг-задачи**:
   * Реализация алгоритмов или функций по заданию.
   * Пример: "Реализуйте алгоритм Дейкстры для поиска кратчайшего пути."
2. **Оптимизация кода**:
   * Улучшение производительности или читаемости существующего кода.
   * Пример: "Оптимизируйте данный алгоритм сортировки, чтобы он работал быстрее."
3. **Отладка кода**:
   * Исправление ошибок в готовом коде.
   * Пример: "В представленном коде поиска подстроки есть ошибка. Найдите и исправьте её."
4. **Тестирование и исправление**:
   * Написание тестов для проверки готового кода.
   * Пример: "Создайте тесты для функции сортировки и найдите её слабые места."

#### **2.3. Платформы для выполнения практических задач**

* **Codeforces, LeetCode, HackerRank** – для подготовки к олимпиадам.
* **Replit, CodePen** – для быстрого выполнения задач в браузере.
* **JetBrains Academy, Codecademy** – интеграция с учебным процессом.

#### **2.4. Критерии оценки**

* **Точность**: работает ли программа правильно для всех тест-кейсов.
* **Скорость выполнения**: успел ли студент решить задачу за установленное время.
* **Качество кода**: Читаемость, соблюдение стиля, отсутствие лишнего кода.
* **Оптимальность**: Использование минимального времени и памяти.

### ****3. Итоговый проект****

#### **3.1. Цель итогового проекта**

* Продемонстрировать применение всех полученных знаний в реальном проекте.
* Развить навыки работы над крупными задачами.
* Создать продукт, который можно использовать на практике или добавить в портфолио.

#### **3.2. Форматы проектов**

1. **Программные приложения**:
   * Разработка приложения для автоматизации задач (например, трекер задач или учёт финансов).
   * Создание веб-приложения с фронтендом и бэкендом.
2. **Алгоритмические решения**:
   * Решение задачи высокой сложности с использованием продвинутых алгоритмов.
   * Пример: Разработка алгоритма планирования маршрутов для логистики.
3. **Проекты с базами данных**:
   * Создание системы учёта данных (например, библиотечная система или CRM).
   * Разработка REST API для работы с базой данных.
4. **Социальные приложения**:
   * Приложение для решения социальной проблемы (например, платформа для волонтёров).

#### **3.3. Этапы работы над проектом**

1. Определение темы и постановка задачи.
2. Разработка прототипа и обсуждение с преподавателем.
3. Реализация проекта с регулярными проверками прогресса.
4. Презентация готового проекта перед комиссией.

#### **3.4. Критерии оценки проекта**

* Полнота реализации: насколько полностью выполнены поставленные задачи.
* Инновационность: Использованы ли новые подходы или технологии.
* Код и документация: Чистота кода, наличие комментариев и технической документации.
* Презентация: Умение чётко рассказать о проекте, продемонстрировать его функционал.

### ****4. Участие в реальных олимпиадах****

#### **4.1. Почему участие в олимпиадах важно?**

* Дает опыт соревнований на высоком уровне.
* Помогает применить полученные знания в реальной среде.
* Улучшает навыки быстрого решения сложных задач.
* Привлекает внимание работодателей и университетов.

#### **4.2. Популярные олимпиады**

* **ICPC (International Collegiate Programming Contest)** – одна из самых престижных олимпиад по программированию.
* **Codeforces Contests, AtCoder, LeetCode Weekly** – регулярные онлайн-соревнования.

#### **4.3. Подготовка к олимпиадам**

1. Решение задач из прошлых лет.
2. Обучение тактикам командного взаимодействия.
3. Участие в тренировочных соревнованиях.

#### **4.4. Поддержка студентов**

* Организация внутренних соревнований для подготовки.
* Предоставление доступа к тренировочным платформам.
* Индивидуальные консультации и анализ ошибок после участия в олимпиадах.

# **Поддержка студентов**

* + Консультации и техническая поддержка.
  + Группы в мессенджерах для обсуждения задач и проектов.

# **Социальные аспекты и вклад**

**1. Участие в реальных проектах**

**1.1. Зачем включать реальные проекты в обучение?**

* Помогает студентам применить теоретические знания на практике.
* Формирует профессиональный опыт, который можно включить в резюме.
* Учит работе в команде и взаимодействию с реальными заказчиками.
* Даёт возможность создавать решения, которые будут использоваться в реальной жизни.

**1.2. Примеры реальных проектов**

1. **Автоматизация бизнес-процессов**:
   * Создание CRM-систем для малого бизнеса.
   * Разработка системы учёта клиентов для небольших компаний.
2. **Образовательные проекты**:
   * Разработка платформы для нужд вуза.
   * Создание приложения для тренировки алгоритмов и подготовки к олимпиадам.
3. **Экологические проекты**:
   * Приложение для мониторинга уровня загрязнения воздуха.
   * Система отслеживания переработки отходов.
4. **Здравоохранение**:
   * Приложение для записи на приём к врачу с интеграцией базы данных.
   * Разработка системы учёта медицинских данных.
5. **Культурные и общественные инициативы**:
   * Приложение для популяризации местного туризма.
   * Создание платформы для волонтёров.

**1.3. Как организовать участие в проектах**

* **Связь с внешними партнёрами**:
  + Коллаборация с НКО, стартапами и малым бизнесом.
  + Привлечение реальных заказчиков для постановки задач.
* **Распределение ролей в команде**:
  + Назначение ролей: разработчики, тестировщики, аналитики, менеджеры.
  + Учёт сильных сторон каждого студента.
* **Этапы работы**:
  + Постановка задачи и сбор требований.
  + Разработка прототипа.
  + Тестирование и доработка.
  + Презентация готового проекта заказчику.
* **Примеры технологий для работы**:
  + Backend: Python (Django/Flask), Node.js.
  + Frontend: React, Angular.
  + Базы данных: PostgreSQL, MongoDB.

**1.4. Результат для студентов**

* Выпускной проект с социальной ценностью.
* Возможность показать проект будущим работодателям.
* Понимание полного цикла разработки программного обеспечения.

**2. Разработка приложений для социальных нужд**

**2.1. Что такое приложения для социальных нужд?**

* Это программные решения, которые решают общественные проблемы или облегчают жизнь определённым группам людей.
* Такие приложения часто создаются для НКО, образовательных учреждений или социальных инициатив.

**2.2. Примеры социальных приложений**

1. **Приложения для людей с ограниченными возможностями**:
   * Приложение для слабовидящих с функцией голосового чтения текста.
   * Система распознавания жестового языка.
2. **Приложения для волонтёров**:
   * Платформа для регистрации и координации волонтёров.
   * Приложение для поиска ближайших волонтёрских мероприятий.
3. **Образовательные приложения**:
   * Приложение для школьников из сельской местности, предоставляющее доступ к учебным материалам.
   * Игровые приложения для обучения детей с особыми образовательными потребностями.
4. **Экологические приложения**:
   * Система расчёта углеродного следа.
   * Приложение для учёта и переработки мусора.
5. **Местные сообщества**:
   * Приложение для улучшения городской среды (например, сбор жалоб на проблемы ЖКХ).
   * Карта доступности городской инфраструктуры для маломобильных граждан.

**2.3. Этапы разработки таких приложений**

1. **Идентификация проблемы**:
   * Проведение опросов или интервью с целевой аудиторией.
   * Сбор информации о потребностях пользователей.
2. **Проектирование решения**:
   * Создание макетов и прототипов интерфейса.
   * Определение функций приложения.
3. **Реализация**:
   * Написание кода и интеграция функционала.
   * Работа с базами данных для хранения информации.
4. **Тестирование**:
   * Проверка работоспособности на реальных пользователях.
   * Исправление ошибок и доработка.
5. **Деплой и поддержка**:
   * Запуск приложения на реальных серверах.
   * Обновление и добавление новых функций на основе обратной связи.

**2.4. Организация работы над социальными приложениями**

* Работа в командах: каждую группу назначают на конкретное приложение.
* Регулярные встречи для обсуждения прогресса.
* Промежуточные презентации для получения обратной связи от преподавателей и целевой аудитории.

**2.5. Результаты разработки**

* Конечное приложение с реальной социальной ценностью.
* Опыт разработки приложений с пользовательским подходом.
* Возможность внести свой вклад в улучшение качества жизни общества.

**3. Итоговые преимущества социальной направленности курса**

* Формирование у студентов чувства социальной ответственности.
* Повышение их вовлечённости в учебный процесс через проекты, которые имеют реальную ценность.
* Создание портфолио проектов, которые можно показать работодателям.
* Вклад в улучшение социальных условий через разработанные решения.

# **Обратная связь и оценка курса**

### 1. ****Сбор предложений и отзывов студентов****

#### 1.1. **Регулярные опросы**

* **Цель**: оценить понимание материала и выявить проблемные зоны.
* **Формат**:
  + Опросники Google Forms или аналогичные инструменты.
  + Вопросы открытого и закрытого типа.
  + Примеры вопросов:
    - Какой модуль/тема оказался самым полезным?
    - Какие аспекты вызвали затруднения?
    - Оцените лекции/практику по 10-балльной шкале.
  + Частота:
    - **Мини-опросы** после каждого модуля.
    - **Итоговый опрос** после завершения курса.

#### 1.2. **Анонимный сбор отзывов**

* **Цель**: получить честные отзывы без страха осуждения.
* **Методы**:
  + Онлайн-анонимные анкеты.
  + Коробка отзывов (в случае очного курса).
  + Вопросы для мотивации:
    - Что вы бы изменили в курсе?
    - Что больше всего понравилось?
    - Чего не хватало?

#### 1.3. **Личные интервью/встречи**

* **Цель**: узнать подробности от наиболее заинтересованных студентов.
* **Формат**:
  + Индивидуальные или групповые беседы.
  + Подготовленный список вопросов + свободная беседа.

#### 1.4. **Мониторинг активности студентов**

* **Цель**: понять, какие темы привлекают больше внимания.
* **Методы**:
  + Анализ активности в LMS (системах управления обучением, например, Moodle).
  + Подсчёт посещаемости занятий и выполнения заданий.

### 2. ****Оценка качества курса****

#### 2.1. **Метрики для оценки**

* **Понимание материала**:
  + Средний балл по тестам и контрольным работам.
  + Уровень решения практических задач (например, успешность на Codeforces).
* **Прогресс студентов**:
  + Сравнение знаний до и после курса (входное и выходное тестирование).
  + Успешность участия в олимпиадах или проектах.

#### 2.2. **Обратная связь от преподавателей/менторов**

* **Цель**: оценить сложность материала и успех методов обучения.
* **Методы**:
  + Регулярные собрания преподавателей.
  + Отчёты по каждому модулю: что удалось, что требует улучшений.

#### 2.3. **Внешняя оценка**

* **Цель**: сравнить курс с аналогами и выявить сильные/слабые стороны.
* **Методы**:
  + Сравнение структуры и содержания с лучшими мировыми программами.
  + Обзор обратной связи от участников сторонних курсов.

### 3. ****Постоянная доработка курса****

#### 3.1. **Анализ собранных данных**

* Обобщение полученных отзывов.
* Выделение ключевых проблем (например, сложность тем, недостаток практики).

#### 3.2. **Внесение изменений**

* Корректировка расписания и времени на модули.
  + Пример: больше времени на динамическое программирование, если оно вызывает трудности.
* Добавление новых тем или технологий.
  + Пример: включение нового фреймворка, если он стал актуальным (например, Svelte вместо React).

#### 3.3. **Тестирование изменений**

* Пилотное внедрение новых методов или тем на небольшой группе студентов.
* Сбор обратной связи об изменениях.

#### 3.4. **Обновление материалов**

* Постоянное обновление слайдов, учебных пособий, видеоуроков.
* Актуализация примеров (например, реальных задач с олимпиад или современных кейсов).

### 4. ****Стратегия вовлечения студентов в улучшение курса****

#### 4.1. **Механизм геймификации**

* + Награды за идеи и предложения, которые внедрены в курс.

#### 4.2. **Создание группы студентов для обратной связи**

* + "Совет студентов", куда входят наиболее активные участники курса.
  + Регулярные встречи для обсуждения улучшений.

#### 4.3. **Организация хакатонов и мастер-классов**

* + Использовать хакатоны как способ протестировать новые подходы.
  + Примеры: решение задач с новыми алгоритмами или работа с новой технологией.

### 5. ****Заключительная обратная связь****

* **Отчёт организаторов**: общий обзор хода курса.
* **Открытая сессия**: обсуждение с участниками после завершения курса.
* **Документированный итоговый отчёт**:
  + Данные об успеваемости.
  + Отзывы студентов.
  + Предложения по доработке для следующих потоков.

# **Итоговые навыки**

После курса студенты смогут:

* + Уверенно решать сложные алгоритмические задачи.
  + Участвовать и побеждать в олимпиадах.
  + Использовать знания для профессиональной разработки.
  + Разрабатывать современные веб- и графические приложения.
  + Работать с современными библиотеками и технологиями.

# **Учебные материалы**

## **Книги по основам языков программирования:**

* + 1. JavaScript: “Учимся кодить на JavaScript”   
       Автор: Джереми Мориц.  
       Ссылка: <https://drive.google.com/file/d/1-lSrs3Xi8IwU-GvuNb72yFjdFXz4SXxF/view>
    2. JavaScript: “JavaScript. Полное руководство. Седьмое издание”  
       Автор: Дэвид Флэнаган.  
       Ссылка: <https://drive.google.com/file/d/1AOjp6fGNgWrlFqNcZPAXnyST1dPIgkLp/view>
    3. JavaScript: “Выразительный JavaScript. Современное веб-программирование. Третье издание”  
       Автор: Марейн Хавербеке.  
       Ссылка: <https://drive.google.com/file/d/1Bd2J309p2OQ8Bpf3Z0oRY-Mw6HgKj_9R/view>
    4. JavaScript: “JavaScript for Web Designers”  
       Автор: Mat Marquis.  
       Ссылка: <https://drive.google.com/file/d/1aq0KjkhftjY1DB3pabxZIxyO4TaJEtAb/view>
    5. JavaSctipt: “Чистый код. Создание, анализ и рефакторинг”  
       Автор: Роберт Мартин.  
       Ссылка: <https://drive.google.com/file/d/1nuDnpK2vTU7xgCsO7OOBJ0-Tex8huyHh/view>
    6. JavaScript: “Как устроен JavaScript”  
       Автор: Дуглас Крокфорд.  
       Ссылка: <https://drive.google.com/file/d/1unuR72Ll2eycqRG5ic0Rm0yQiS4MbQ41/view>
    7. Python: “Python. Погружение в математику с Minecraft”   
       Автор: А.В. Корягина и А.В. Корягиной.  
       Ссылка: <https://drive.google.com/file/d/1DZ_ypoBzz2wGzwEd3cwRx4nKdQ4ObxnX/view>
    8. Python: “Программирование на Python в примерах и задачах”  
       Автор: Васильев А. Н.  
       Ссылка: <https://drive.google.com/file/d/1Feimt2Ls1aPEKIM3IHnsq2d2cyCcES_m/view>
    9. Python: “Изучаем Python: программирование игр, визуализация данных, веб-приложения”  
       Автор: Эрик Мэтиз.  
       Ссылка: <https://drive.google.com/file/d/1xg0jx5pR5HgRePT9H4sNMWs1iRsuDaLd/view>
    10. Python: “Простой Python. Современный стиль программирования, 2-е издание”  
        Автор: Любанович Б.  
        Ссылка: <https://drive.google.com/file/d/1PgagptxWmyrYW9rTbTARTuiZrIS_dwc_/view>
    11. Python: “Однострочники Python: лаконичный и содержательный код”  
        Автор: Майер Кристиан.  
        Ссылка: <https://drive.google.com/file/d/1uJ7yxEwvbrVdLaToPdbBe9rI0OAn08qn/view>
    12. Python: “Большая книга проектов Python”  
        Автор: Свейгарт Эл.  
        Ссылка: <https://drive.google.com/file/d/1MriEYlXEjZ9Ud3NDFXP4RSfVq65oPrRZ/view>
    13. C++: “Думай как программист”   
        Автор: Антон Спрол.  
        Ссылка: <https://drive.google.com/file/d/1CB7ewLBDWRA5MCPfTUDBlR1L1JqR5WJF/view>
    14. С++: “Изучаем программирование игр”  
        Автор: Майкл Доусон.  
        Ссылка: <https://drive.google.com/file/d/1brTApBYCb7pxxFn0JEC8T2cRf9FX7eNy/view>
    15. С++: “Язык программирования C++”  
        Автор: Бьерн Страуструп.  
        Ссылка: <https://drive.google.com/file/d/1K_rHpiTEtNWZUzIXHkqSJV5kQnmA-JSo/view>
    16. С++: “Эффективный и современный C++: 42 рекомендации по использованию C++11 и C++14”  
        Автор: Скотт Мейерс.  
        Ссылка: <https://drive.google.com/file/d/1oyvEZImZurEaGDALphz7_3WIiwPQCoQd/view>
    17. С++: “Современный C++. Безопасное использование”  
        Автор: Джон Лакос.  
        Ссылка: <https://drive.google.com/file/d/1aJtlJtj-t5A95UTy8D4AMupQZmk_J88u/view>
    18. C++: “Qt 6. Разработка оконных приложений на C++”  
        Автор: Николай Прохоренок.  
        Ссылка: <https://drive.google.com/file/d/1bDjEwbGD13yk-NozmTXEesqJjDDUkmiE/view>
    19. Java: “Head First Java”  
        Автор: Кэти Сиерра и Берт Бейтс.  
        Ссылка: <https://drive.google.com/file/d/1-NgmAm7_hy3xSTJuxED6ZkfzFbyH4ZGa/view>
    20. Java: “Java для чайников”  
        Автор: Барри Берд  
        Ссылка: <https://drive.google.com/file/d/18E5A7TYgOx4-OT8dQiwa1pn0RSwdJLHZ/view>
    21. Java: “Java для всех”  
        Автор: Алексей Васильев.  
        Ссылка: <https://drive.google.com/file/d/1z4bd2ChU8hqGB9howa80WP-2ocsc7BON/view>
    22. Java: “Think Java: How to Think Like a Computer Scientist”  
        Автор: Аллен Дауни, Крис Мэйфилд.  
        Ссылка: <https://drive.google.com/file/d/14gDdA3NwQa00_MAAiWVlfMntyferQSNw/view>
    23. Java: “Java. Руководство для начинающих”  
        Автор: Герберт Шилдт.  
        Ссылка: <https://drive.google.com/file/d/10jj83ed9tFQp1SryXhzI31urObI8uEJU/view>

## **Книги по веб-разработке:**

* + 1. “HTML и CSS: Разработка и дизайн веб-сайтов”  
       Автор: Джон Дакетт.  
       Ссылка: <https://drive.google.com/file/d/1jvP-fLDAXG8KW9lVAnBBDOwbts3uLcyu/view>
    2. “Изучаем HTML5”  
       Автор: Брюс Лоусон, Реми Шарп.  
       Ссылка: <https://drive.google.com/file/d/1sQbfUcFoQXF6OTRY4dW_UU5DJ_lbah33/view>
    3. “CSS: Секреты”  
       Автор: Лиа Веру.  
       Ссылка: <https://drive.google.com/file/d/1g2qbotcVpOsbFpqsRHoqVEvsVk0gMwpv/view>
    4. “CSS: Каскадные таблицы стилей для профессионалов”  
       Автор: Эрик Мейер.  
       Ссылка: <https://drive.google.com/file/d/1Mk1Ir7mnQGg9su1NkFzQp-Uy_8JqQbFx/view>
    5. “HTML5: Карманный справочник”  
       Автор: Дженнифер Роббинс.  
       Ссылка: <https://drive.google.com/file/d/1LwjTVWWjd8Wr5yppt79ShOB-esbAZY9I/view>
    6. “Изучаем XML”  
       Автор: Эрик Т. Рей.  
       Ссылка: <https://drive.google.com/file/d/1jIEw2ij9HbVm6OmmI5cXt4qLUDiTB6TE/view>
    7. “Become a ninja with Angular”  
       Автор: Ninja Squad.  
       Ссылка: <https://drive.google.com/file/d/1o91yBWs5mEXMeV5FqzmJO7kP2CWNbtTl/view>
    8. “Vue.js: Up and Running”  
       Автор: Калеб Портс.  
       Ссылка: <https://drive.google.com/file/d/1G63ip-6sIndzl5ZreaPL0sSNNghPzz2d/view?usp=drive_link>
    9. “Vue.js 2 Web Development Projects”  
       Автор: Mike Street, Andrea Passaglia, Paul Halliday.  
       Ссылка: <https://drive.google.com/file/d/19-3EYt7p9hqnhqxN22VZesdK4QyPdtbJ/view>
    10. “Vue.js in Action”  
        Автор: Эрик Хэнчетт.  
        Ссылка: <https://drive.google.com/file/d/1i4TdkE1i9UJBupsUa7az1My1Hw4VNXfv/view?usp=drive_link>
    11. “React.js. Быстрый старт”  
        Автор: Stoyan Stefanov.  
        Ссылка: <https://drive.google.com/file/d/169CJxcDJNdMaPotdkv1CmChImhXI0wkh/view>
    12. “Учебный курс по React”  
        Ссылка: <https://drive.google.com/file/d/19JHlToQYsfq5tRC8DIri_OlmqyVBfwmo/view>
    13. “Pro React 16”  
        Автор: Adam Freeman.  
        Ссылка: <https://drive.google.com/file/d/1DWEvQQLsARWFg7wRUmWEe5kJzhmdgq2w/view>
    14. “The Road to React”  
        Автор: Robin Wieruch.  
        Ссылка: <https://drive.google.com/file/d/1ZZb0DwiLTlL-IyLpVbH6WTKZMtH8BvBb/view>  
        <https://github.com/the-road-to-learn-react/the-road-to-learn-react-russian>
    15. “Django. Разработка веб-приложений на Python”  
        Автор: Джефф Форсье, Пол Биссекс, Уэсли Дж. Чан.  
        Ссылка: <https://drive.google.com/file/d/1xfkpvMUxkgNP5PTcyAz_aLtvN7LUi7eP/view?usp=drive_link>
    16. “Django: практика создания Web-сайтов на Python”   
        Автор: Владимир Дронов.  
        Ссылка: <https://drive.google.com/file/d/1xfkpvMUxkgNP5PTcyAz_aLtvN7LUi7eP/view>
    17. “REST APIs with Django: Build powerful web APIs with Python and Django”  
        Автор: William S. Vincent.  
        Ссылка: <https://drive.google.com/file/d/1HYhhrj3eXHKyC7aJZJBw6p-Upi-zrZK1/view>
    18. “Agile Web Development with Rails 5”  
        Автор: Сэм Руби.  
        Ссылка: <https://drive.google.com/file/d/1u8ZXFNnDmYd_XifjJmar6gsUVrPKVd5w/view>
    19. “Ruby on Rails для начинающих”  
        Автор: Майкл Хартл.  
        Ссылка: <https://drive.google.com/file/d/1yA9HeX24f4uo_9tYRbNgq-62q9wjNHOR/view>
    20. “Ruby. ООП.”  
        Автор: Сэнди Метц.  
        Ссылка: <https://drive.google.com/file/d/1rzfKNw0Iuy61p4zGUlnbG-4zRmQppeez/view>
    21. “Laravel. Полное руководство”  
        Автор: Мэтт Стаффер.  
        Ссылка: <https://drive.google.com/file/d/1sU8TGklD8WGEwIMxIDqaQ14B9g6X584x/view>
    22. “Laravel 8. Быстрая разработка веб-сайтов на PHP”  
        Автор: Владимир Александрович Дронов.  
        Ссылка: <https://drive.google.com/file/d/1skroG6tDLBBn3c4OuO02YRD_TeYmcxcx/view>
    23. “Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript”  
        Автор: Браун Итан.  
        Ссылка: <https://drive.google.com/file/d/1WKuul33hAPhZvmnBttqSW0VN_qmblt89/view>
    24. “Node.js в действии”  
        Автор: Алекс Янг, Майк Кантелон.  
        Ссылка: <https://drive.google.com/file/d/1t5W6x5e0B2-8sNDToY4EBRYT4gZsc9iJ/view>
    25. “Руководство по Node.js”  
        Ссылка: <https://drive.google.com/file/d/1XyMDkf0EtKJypQYA2d34GipAHH6Fkca9/view>
    26. “Изучаем Node. Переходим на сторону сервера”  
        Автор: Шелли Пауэрс.  
        Ссылка: <https://drive.google.com/file/d/1LqIm54YXs31mQVvVwARQP6O6v0ygsTy6/view>
    27. “Flask Web Development”  
        Автор: Miguel Grinberg.  
        Ссылка: <https://drive.google.com/file/d/1y9Sn6tIW7PaUjz4oM3IKA1BIsAhbORMP/view>
    28. “Flask: веб-разработка капля за каплей”  
        (Документация)  
        Ссылка: <https://drive.google.com/file/d/19D4zRw2nzr1XH014eMaAlJ9AETZeQOHs/view>
    29. “Building Web Applications with Flask”  
        Автор: Italo Maia.  
        Ссылка: <https://drive.google.com/file/d/1B5mkoDdKTyoOOTNrMLZBAkIQhW0VU0vb/view>
    30. “Flask Framework Cookbook”  
        Автор: Shalabh Aggarwal.  
        Ссылка: <https://drive.google.com/file/d/1lir4aLTupuc9oJd3Ezokkp0oCkInHEp_/view>

## **Книги по Алгоритмам:**

* + 1. “Грокаем алгоритмы”   
       Автор: Адитья Бхаргава.   
       Ссылка: <https://drive.google.com/file/d/1R3dNbmJKILBV4-X_6N9Lucx_jp6BkkSN/view>
    2. “Совершенный алгоритм. Основы”   
       Автор: Тим Рафгарден.  
       Ссылка: <https://drive.google.com/file/d/1zRz3n4Uz34WfAoj1hVxnwNjrT1U3A21_/view>
    3. “Совершенный алгоритм. Графовые алгоритмы и структуры данных”   
       Автор: Тим Рафгарден.  
       Ссылка: <https://drive.google.com/file/d/10j3HdoBeABgM-RYuy1LMtrDSDeBx3UFR/view>
    4. “Алгоритмы для начинающих”   
       Автор: Луридас Панос.   
       Ссылка: <https://drive.google.com/file/d/1qDi78X9ZiUiUctCn2sJTzlE5sii_WcyP/view>
    5. “Введение в анализ алгоритмов”   
       Автор: Майкл Солтис.  
       Ссылка: <https://drive.google.com/file/d/1Xsxu42PiUahtc_R0teOSl2nEVosYdkx5/view>
    6. “Алгоритмы: разработка и применение”   
       Автор: Джон Клейнберг, Эва Тардос.   
       Ссылка: <https://drive.google.com/file/d/1b7ftZawwcnw_cKRvZgk31Ebip_en1HQW/view>
    7. “Алгоритмы. Руководство по разработке”   
       Автор: Стивен Скиена.   
       Ссылка: <https://drive.google.com/file/d/1d1P83gb_LMk6FmJFGlMWvJAEUCC-GFW6/view>
    8. “Алгоритмические трюки для программистов”  
       Автор: Генри Уоррен.   
       Ссылка: <https://drive.google.com/file/d/1whv-BoT1Jwz2G0A04yOawwjghHDEtOa9/view>
    9. “Алгоритмы. Справочник с примерами на C, C++, Java и Python”   
       Автор: Джордж Хайнеман, Гари Поллис, Стэнли Селков.   
       Ссылка: <https://drive.google.com/file/d/1ZMAu_66ELvR5GnBsnQR9Nm8s06B6UFOu/view>
    10. “Алгоритмы на C++”   
        Автор: Роберт Седжвик.   
        Ссылка: части 1-4: <https://drive.google.com/file/d/17SX0HaF5KKeRpa9gwz-LErpRGdEeaqtz/view>  
        Часть 5: <https://drive.google.com/file/d/1x7F3gM1SPtB-e610EnTOtL0VkiCu-duD/view>
    11. “Алгоритмы шифрования. Специальный справочник” Автор: Сергей Панасенко.   
        Ссылка: <https://drive.google.com/file/d/1btSoqikkez_QnBjM3-2tE-Vl30zdHktY/view>

## **Книги по базам данных:**

* + 1. “Изучаем SQL”  
       Автор: Алан Болье.  
       Ссылка: <https://drive.google.com/file/d/1wH6X2yUny11BC9zvGfyd_ZyQqla88uIP/view>
    2. “SQL для чайников”   
       Автор: Аллен Тейлор.  
       Ссылка: <https://drive.google.com/file/d/1EFlp1M1Aw3fhuLcba4IRiTINz9A6Q6uC/view>
    3. “SQL: быстрое погружение”  
       Автор: Уолтер Шилдс.  
       Ссылка: <https://drive.google.com/file/d/1e3t1Tp3JI7orHwPAcYs4SPZW4KqjQ2U-/view>
    4. “SQL. Сборник рецептов”  
       Автор: Энтони Молинаро, Роберт де Грааф.  
       Ссылка: <https://drive.google.com/file/d/1oVLzycOZsndcX5wfr1pyFqFiihDdN_p5/view>
    5. “PHP и MySQL. 25 уроков для начинающих”  
       Автор: В. Дронов.  
       Ссылка: <https://drive.google.com/file/d/11qMn6uUNvheXONmEmQhk4kixuMzCzF18/view>
    6. “MySQL. Сборник рецептов”  
       Автор: Поль Дюбуа  
       Ссылка: <https://drive.google.com/file/d/1EN4q6M403gwbaPJxltAztmXZUwNNJ8QY/view>
    7. “Изучаем PostgreSQL 10”   
       Автор: С. Джуба, А. Волков.  
       Ссылка: <https://drive.google.com/file/d/11jhWXbePhGvd9j4deqVYk7c57rl5vtXz/view>
    8. “Оптимизация запросов PostgreSQL”   
       Автор: Г. Домбровская, Б. Новиков, А. Бейликова  
       Ссылка: <https://drive.google.com/file/d/1xqkZNK-EJ8wFPcg4XH-0UtOYU-D6X69X/view>
    9. “Основы технологий баз данных: учебное пособие”  
       Автор: Б. Новиков, Е. Горшкова, Н. Графеева.  
       Ссылка: <https://drive.google.com/file/d/1XsLvDnsDkgMHwaHGz3pZ6Zn8sNudycPF/view>
    10. “MongoDB в действии”  
        Автор: Кайл Бэнкер.  
        Ссылка: <https://drive.google.com/file/d/1PYOB1lRVkNmSpIpOjX5Z3hovfjyptpyh/view>

## **Онлайн-ресурсы:**

* + 1. Codeforces  
       Ссылка: <https://codeforces.com/>
    2. LeetCode  
       Ссылка: <https://leetcode.com/>
    3. AtCoder  
       Ссылка: <https://atcoder.jp/>.
    4. Metanit  
       Ссылка: <https://metanit.com/>
    5. Doka  
       Ссылка: <https://doka.guide/>
    6. WebReference  
       Ссылка: <https://webref.ru/>
    7. Python  
       Ссылка: <https://www.python.org/>