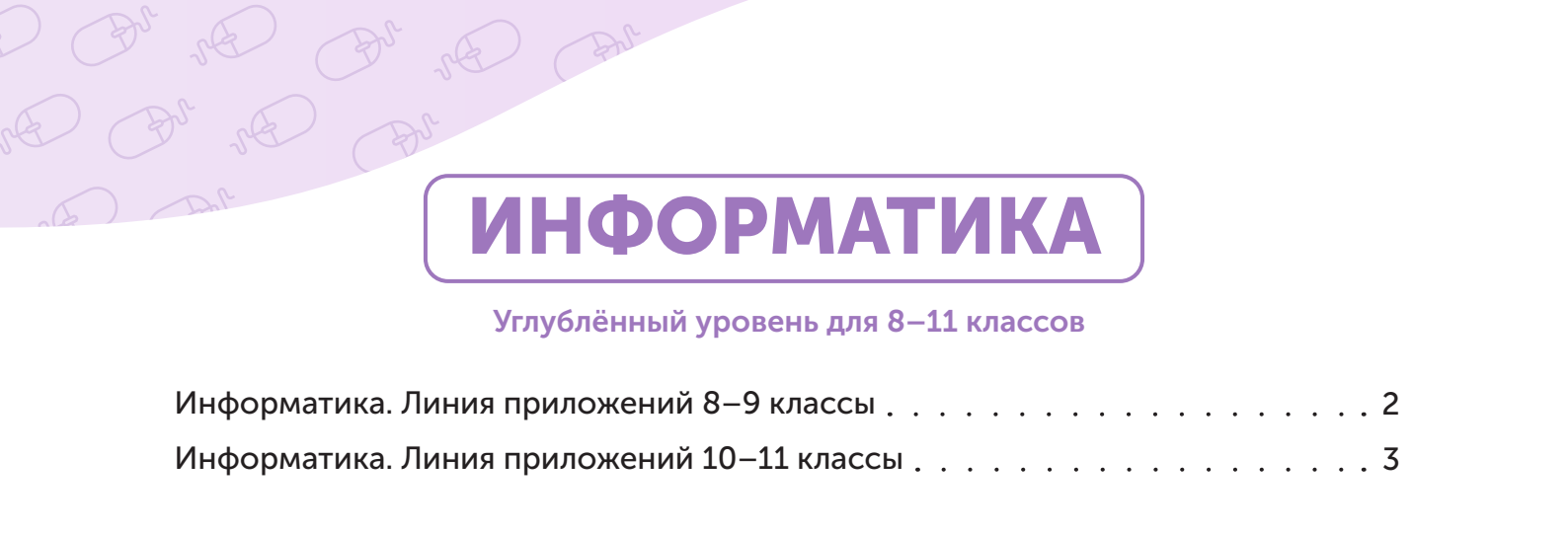




ИНФОРМАТИКА

Углублённый уровень для 8–11 классов

Информатика. Линия приложений 8–9 классы 2

Информатика. Линия приложений 10–11 классы 3



ИНФОРМАТИКА. ЛИНИЯ ПРИЛОЖЕНИЙ

8–9 классы

- Архитектура компьютерных игр. Установка и инициализация библиотеки Pygame
- Игровое окно, координатная сетка и бесконечный игровой цикл
- Работа с цветом (модель RGB). Отрисовка простейших геометрических фигур
- Понятие FPS. Управление временем и скоростью игры
- Обработка базовых событий: закрытие окна, нажатие клавиш клавиатуры
- Создание программы с управляемой геометрической фигурой
- Загрузка растровых изображений. Понятие прозрачности (Alpha-канал)
- Класс Sprite в Pygame. Создание и группировка игровых объектов
- Анимация персонажа методом быстрой смены кадров
- Программирование движения объектов по экрану с клавиатуры и мыши
- Практикум: реализация плавного перемещения анимированного персонажа
- Понятие хитбоксов
- Создание простейших соперников
- Работа со звуком: загрузка и воспроизведение фоновой музыки
- Практикум: создание анимации шарика, передвигающегося по экрану
- Практикум: ловля предмета
- Эффект частиц: генерация и анимация простейшей системы частиц
- Мини-игра «Арканоид»
- Модель «Уклонение от препятствий»
- Простейшая модель Tower Defense
- Модель «Стрельба по движущейся цели»
- Модель «Полёт с препятствиями»
- Модель «Попади в цель»
- Создание собственной игры. Подготовка итогового проекта (2 занятия)
- Защита итогового проекта
- Релейный регулятор
- PD (движение по линии)
- Движение по инверсной линии
- объезд препятствий
- Маршруты
- Кегельринг
- Точные движения по энкодеру
- Точные движения по гироскопу
- RGB
- HSV
- Движение в зависимости от цвета
- Лабиринт
- Обход лабиринта с линиями
- Штрих-код
- Сетка линий
- Движение змейкой внутри сетки линий
- Картографирование
- Локализация
- Решение комплексных задач (2 занятия)



ИНФОРМАТИКА. ЛИНИЯ ПРИЛОЖЕНИЙ

10–11 классы

- Введение в Git: установка, настройка. Создание репозитория
- Основные команды: `init`, `add`, `commit`, `status`, `log`
- Работа с удалённым репозиторием: `push`, `pull`, `clone` (GitHub)
- Ветвление: создание ветки, `merge`, разрешение конфликтов
- Базы данных. Введение в SQLite. Создание таблиц, типы данных
- CRUD-операции: `INSERT`, `SELECT`, `UPDATE`, `DELETE` через `sqlite3`
- Проектирование БД для приложения (нормализация, связи)
- SQLAlchemy ORM: создание моделей, сессии
- Добавление, запросы, фильтрация с использованием SQLAlchemy
- Обновление и удаление записей через ORM
- Юнит-тестирование: библиотека `unittest`. Структура тестового класса
- Написание тестов для функций и классов (`assertEqual`, `assertTrue`)
- Тестирование с использованием фикстур (`setUp`, `tearDown`)
- Интеграционное тестирование: тестирование работы с БД
- Практикум: написание тестов для класса «Банковский счёт»
- Запуск тестов в CI (обзор GitHub Actions)
- Введение в веб-фреймворки. Установка Flask. Маршруты (`routes`)
- Шаблоны Jinja2: передача переменных, условные операторы, циклы
- Формы: GET- и POST-запросы. `request.form`, `request.args`
- Создание простого сайта-визитки с несколькими страницами
- Работа с внешними API: библиотека `requests`. GET-запросы
- Получение данных о погоде, курсе валют, новостей
- Обработка JSON-ответов. Отображение данных в шаблоне
- Создание собственного API на Flask: возврат JSON, маршруты с параметрами
- Методы POST, PUT, DELETE в API. Тестирование через Postman
- Проект: «Умная панель жителя города» — архитектура и интеграция Bootstrap
- Работа с внешними API и шаблонами Jinja2
- Формы, база данных и кэширование
- Создание собственного API и логирование
- Тестирование через Postman, юнит-тесты и GitHub Actions
- Деплой приложения на бесплатный хостинг (Render, PythonAnywhere)
- Чат-бот для VK
- Гибкие методологии: Scrum, Kanban. Роли, артефакты
- Выбор темы проекта. Составление технического задания
- Проектирование архитектуры: клиент-сервер, БД, схема данных
- Продвинутый Git: ветвление по модели Git Flow
- Документирование кода (`docstrings`, README)
- Разработка бэкенда на Flask: модели данных
- Реализация REST API для работы с задачами (CRUD)
- Разработка фронтенд-части: простые HTML-шаблоны с формами
- Разработка фронтенд-части: подключение Bootstrap и реализация всех форм
- Валидация данных, обработка ошибок, возврат статус-кодов
- Интеграция внешнего API
- Создание тестов для API и приложения
- Docker: установка, образы, контейнеры. Dockerfile
- Docker Compose: описание многоконтейнерных приложений (`app + db`)