

**Методические указания для выполнения курсовой работы по дисциплине
«Основы алгоритмизации и программирования»**

Цель работы:

Методическое указание содержит требования к содержанию и оформлению пояснительной записи, методические указания к выполнению требований к технической документации в виде приложений, а также требования к процедуре защиты курсового проекта.

Предназначены для студентов 2 курса по специальностям **ПОВТ; ПИ; АСОИ; ТОС**.

ОГЛАВЛЕНИЕ

- 1. Введение**
- 2. Организационно-методические указания**
 - 1. Цели и задачи курсовой работы**
 - 2. Основные этапы выполнения курсовой работы**
 - 3. Методические указания по структуре и содержанию курсовой работы**
 - 4. Разбор типовой структуры и содержания курсовой работы**
 - 5. Требования и правила изложения текстового материала**
 - 6. Порядок защиты курсовой работы**
 - 7. Критерии оценивания курсовой работы**
 - 8. Темы курсовых работ**

ВВЕДЕНИЕ

Цель работы:

Курсовая работа направлена на развитие практических навыков программирования, проектирования и разработки программных продуктов с

использованием языка программирования Python. Основное внимание уделяется следующим задачам:

- изучение алгоритмизации и основных подходов к решению задач;*
- закрепление знаний, полученных в процессе теоретического изучения программирования;*
- практическое применение принципов работы с данными, создания алгоритмов и проектирования пользовательских интерфейсов;*
- подготовка к профессиональной деятельности.*

1. Организационно-методические указания

Курсовая работа по дисциплине «Основы алгоритмизации и программирования» предназначена для закрепления теоретических знаний и получения практических навыков программирования. Студенты выполняют проект с использованием технологий Python для серверной части, JavaScript для клиентской части и хранение данных в работе с базами данных.

Основное внимание уделяется созданию завершенного приложения, которое включает:

- Логику работы с данными на серверной стороне;*
- Разработку пользовательского интерфейса;*
- Организацию базы данных.*

Работа выполняется индивидуально. Курсовой проект должен быть завершен и защищен до сдачи экзамена по дисциплине.

2. Методические указания по структуре и содержанию курсовой работы

Требования к объему

- Текст работы: 30 (без приложений).*
- Работа должна содержать текстовое описание, схемы, рисунки, скриншоты.*

Структура курсовой работы

Курсовая работа состоит из следующих разделов:

1. Титульный лист

- *Оформляется по установленному образцу.*

2. Оглавление

- *Указывается структура работы с номерами страниц.*

3. Введение

- *Обоснование актуальности темы, цели и задачи работы.*

4. Основная часть

- *Аналитическая часть: Анализ предметной области, формулировка требований.*
- *Практическая часть: Подробное описание разработки приложения:*
 - Логика обработки данных на серверной стороне;*
 - Разработка интерфейса пользователя;*
 - Проектирование базы данных.*

5. Заключение

- *Анализ результатов, выводы и предложения по улучшению.*

6. Список использованной литературы

- *Перечень источников, оформленный по стандарту.*

7. Приложения

- *Дополнительные материалы (коды программ, результаты тестов, графики).*

3. Разбор типовой структуры и содержания курсовой работы

Введение

- *Актуальность выбранной темы.*
- *Цели и задачи курсовой работы.*

- Краткое описание практического значения разработки.

Аналитическая часть

- Анализ существующих решений в рамках темы.
- Формулировка требований к разрабатываемому приложению.
- Описание структуры и функций разрабатываемой системы.

Практическая часть

- Серверная часть:
 - Реализация бизнес-логики с использованием Python;
 - Функционал обработки данных (например, работа с файлами).
- Клиентская часть:
 - Разработка интерфейса пользователя на JavaScript;
 - Реализация взаимодействия с серверной частью через файлы или другие механизмы.
- База данных:
 - Проектирование структуры базы данных;
 - Реализация запросов для обработки данных.

Заключение

- Анализ достижения целей.
- Результаты тестирования.
- Выводы и предложения.

ПОРЯДОК ЗАЩИТЫ КУРСОВОЙ РАБОТЫ

Курсовая работа за пять дней до защиты в электронной форме предоставляется преподавателю для рецензии. Преподавателем пишется рецензия на курсовой проект, в случае большого количества недоработок курсовой проект возвращается на доработку и вновь предоставляется на рецензию.

Курсовой проект может быть снят с защиты в следующих случаях:

- содержание не соответствует теме курсового проекта;
- работа переписана с одного или нескольких источников (в том числе из сети Интернет) более чем на 50%;

В этом случае студенту назначается новая тема курсового проекта.

Защита курсовой работы проходит в открытой форме (на защите могут присутствовать студенты и преподаватели). Преподавателем в соответствии с календарным графиком выполнения курсового проектирования составляется календарный график защиты курсового проектирования.

Материалы, предоставляемые к защите

Для защиты курсовой работы студент должен предоставить следующие материалы:

- Пояснительная записка.
- Доклад и презентация по теме курсовой работы.
- Результаты курсовой работы на внешнем носителе (разработанный проект, пояснительная записка, презентация).

Требования к структуре и содержанию слайдов к защите

Для защиты курсовой работы необходимо предоставить слайды, созданные в пакете PowerPoint (*.ppt); Canva (*.ppt); или Figma (*.ppt); следующего состава:

- Титульный слайд – указывается наименование курсовой работы, тема курсовой работы, ФИО студента, ФИО руководителя.
- Задание на курсовую работу.
- Слайды, описывающие содержание курсовой работы (количество и содержание слайдов определяется руководителем).
- Выводы и заключения.
- Разработанный проект

Процедура защиты курсовой работы следующая:

- изложение автором содержания проекта в течение 5-10 мин (доклад) с демонстрацией презентации;
- вопросы преподавателя по содержанию работы и ответы на них студента;
- оглашение оценки за курсовую работу по четырех балльной системе:

«отлично», «хорошо», «удовлетворительно», «неудовлетворительно».

В докладе, который студент должен представить кратко и четко, необходимо отразить:

- цели и задачи курсового проекта;
- характеристику объекта, на материалах которого выполнен проект;
- содержание аналитической и проектной частей с обоснованием принятых решений; особо следует останавливаться на новых, оригинальных решениях (при их наличии в проекте).

В докладе не следует уделять много времени на пояснение общезвестных положений. Курсовая работа должна быть защищена до сдачи экзамена по междисциплинарному курсу «Основы алгоритмизации и программирования» (PYTHON)

Студенты, не сдавшие курсовые работы или получившие на защите неудовлетворительные оценки, не допускаются к очередным экзаменам.

6. КРИТЕРИИ ОЦЕНИВАНИЯ КУРСОВОЙ РАБОТЫ

Критериями оценки курсовой работы являются:

Ориентировочными критериями для выставления отметки за курсовую работу могут являться:

- соблюдение сроков выполнения и сдачи курсовой работы;
- внешний вид и правильность оформления курсовой работы;
- обоснование актуальности курсовой работы;
- корректность формулировки характеристик исследования (проблемы, объекта, предмета, задач и т.п.)
 - соответствие содержания работы заявленной теме исследования;
 - полнота раскрытия темы исследования;
 - завершенность и полнота решения всех задач, поставленных перед исследованием;
 - взаимосвязь теоретического и практического материала;
 - наличие в тексте сносок и гиперссылок;
 - наглядность и правильность оформления иллюстративного материала;
 - наличие и качество приложений;
 - правильность оформления списка литературы;
- глубина теоретического анализа, умение разобраться в основных проблемах заданной темы, знание и понимание основных точек зрения и дискуссионных проблем;
- связь работы с жизнью, с практической действительностью;
- умение делать выводы;
- качество введения и заключения;
- самостоятельность изложения, творческий подход к рассматриваемой проблеме, умение излагать и аргументировать свою точку зрения;

- логичность и грамотность изложения материала, владение терминологией и стилем научного изложения;
- отсутствие содержательных ошибок принципиального характера;
- теоретическая и практическая ценность работы (при необходимости);
- наличие и полнота описания практической апробации;
- качество оформления работы.

Отметка «отлично» выставляется при соблюдении всех требований к курсовой работе и выполнении курсовой работы в установленные сроки.

Отметка «хорошо» выставляется, если при наличии выполненной на высоком уровне реферативной части, исследовательская часть и выводы недостаточно убедительны.

Отметка «удовлетворительно» выставляется при частичном соблюдении требований к курсовой работе: суть проблемы раскрыта недостаточно тщательно; отсутствует одна из структурных частей работы; работа неправильно оформлена. Отметка «неудовлетворительно» выставляется, если не соблюdenы все основные требования к курсовой работе, в частности: при ее написании использовалось малое количество источников, притом устаревших, литературной основой являлись только учебники или научно-популярная литература; в работе искажены научные положения.

Для выполнения курсовой работы по дисциплине предлагается следующий перечень тем. Студенты могут выбрать одну из предложенных тем или предложить свою с согласия преподавателя.

Темы по курсовой работе, темы детализированы с точки зрения описания, технического задания и плана выполнения, чтобы облегчить реализацию курсового проекта. Каждая тема предполагает использование Python для серверной логики, JavaScript для пользовательского интерфейса и хранения данных в базе данных.

Тема 1: Интерактивная система тестирования знаний по информатике

Описание: Разработка веб-приложения, позволяющего студентам проходить тесты по темам курса «Основы алгоритмизации и программирования». Система должна поддерживать создание тестов преподавателями, автоматическую проверку и отображение результатов.

Техническое задание:

- Серверная часть (*Python, FastAPI*):
 - Регистрация и авторизация пользователей (студенты, преподаватели).
 - CRUD-функциональность для тестов и вопросов.
 - Проверка ответов и ведение статистики прохождений.
- Клиентская часть (*JavaScript*):
 - Удобный интерфейс прохождения тестов.
 - Панель преподавателя для создания и редактирования тестов.
 - Отображение статистики и результатов.

План выполнения:

1. Сбор требований, описание ролей пользователей.
2. Проектирование структуры базы данных (*users, tests, questions, results*).
3. Реализация REST API для взаимодействия с тестами и результатами.
4. Создание интерфейса для прохождения тестов и управления ими.
5. Проведение юзабилити-тестирования и написание пояснительной записи.

Тема 2: Система управления расписанием преподавателей (Teacher Timetable Manager)

Описание: Создание веб-приложения для составления и управления индивидуальными расписаниями преподавателей. Студенты могут бронировать встречи на основе свободных слотов преподавателя.

Техническое задание:

- Серверная часть (*Python, FastAPI*):
 - Регистрация преподавателей и студентов.
 - Управление расписанием, хранение доступных слотов.
 - API для бронирования, подтверждения и отмены встреч.
- Клиентская часть (*JavaScript*):
 - Интерактивный календарь для выбора времени.
 - Панель преподавателя для настройки доступности.
 - Уведомления и подтверждение бронирований.

План выполнения:

1. Анализ требований и пользовательских сценариев.
2. Создание моделей расписания, встреч и уведомлений.
3. Реализация API управления расписанием.
4. Разработка интерфейса календаря и системы бронирования.
5. Финальное тестирование и написание документации.

Тема 3: Онлайн-бронирование билетов в кинотеатр **Описание:** Веб-сервис, позволяющий пользователям бронировать билеты в кинотеатр с выбором фильма, сеанса и мест. Администрация может управлять расписанием показов.

Техническое задание:

- Серверная часть (*Python, FastAPI*):
 - Модели кинотеатров, залов, фильмов, сеансов, мест и бронирований.
 - Регистрация пользователей, управление бронированиями.
- Клиентская часть (*JavaScript*):
 - Афиша фильмов и выбор сеансов.
 - Схема зала с выбором мест.
 - Уведомления о подтверждении брони.

План выполнения:

1. Разработка модели данных для фильмов, залов и бронирований.
2. Реализация API для фильтрации сеансов и мест.
3. Интерфейс выбора фильма, времени и мест на схеме зала.
4. Интеграция функций регистрации и управления бронированиями.
5. Презентация проекта и финальное тестирование.

Тема 4: Платформа визуализации алгоритмов сортировки **Описание:** Обучающее приложение, позволяющее пользователю наблюдать пошаговую визуализацию работы алгоритмов сортировки: пузырьковая, выбором, вставками и быстрая сортировка.

Техническое задание:

- Серверная часть (*Python, FastAPI*):
 - Хранение пользовательских настроек (скорость, размер массива).
 - Выдача конфигурации массива для сортировки.
- Клиентская часть (*JavaScript*):
 - Анимация сортировки с возможностью управления скоростью.
 - Выбор типа алгоритма, шаг за шагом.

План выполнения:

1. Подготовка анимационного движка на *JavaScript*.
2. Реализация серверной части для выдачи массивов и настроек.
3. Интеграция интерфейса и визуализации.
4. Проведение серии тестов и оптимизация производительности.
5. Составление отчета и подготовка к защите.

Тема 5: Учебный портал по Python с автопроверкой решений **Описание:** Платформа для обучения Python с теоретическим материалом, задачами, встроенным редактором кода и автоматической проверкой решений студентов.

Техническое задание:

- **Серверная часть (Python, FastAPI):**
 - Авторизация студентов.
 - Хранение уроков, задач, тестов и решений.
 - Проверка кода с ограничением по времени и памяти.
- **Клиентская часть (JavaScript):**
 - Навигация по темам курса.
 - Встроенный редактор кода с отправкой на проверку.
 - Отображение результата и ошибок.

План выполнения:

1. Проектирование структуры уроков и задач.
2. Разработка системы автотестов и проверки кода.
3. Создание интерактивного редактора решений.
4. Система прогресса, достижений и уровней.
5. Финальная сборка и подготовка презентации.

Тема 6: Визуальный конструктор блок-схем **Описание:** Веб-приложение для построения блок-схем, отражающих алгоритмы, с возможностью сохранения, редактирования и экспорта. Может использоваться в учебных целях для визуального представления логики программ.

Техническое задание:

- **Серверная часть (Python, FastAPI):**
 - Регистрация и авторизация пользователей.
 - Хранение схем и метаданных пользователей.
 - API для загрузки, сохранения и удаления схем.
- **Клиентская часть (JavaScript):**
 - Drag-and-drop редактор блоков (ввод, процесс, условие, конец).
 - Возможность соединения блоков стрелками.
 - Экспорт в изображение (PNG, SVG) и PDF.

План выполнения:

1. Проектирование структуры блок-схем и связей.
2. Реализация визуального редактора.
3. Настройка серверной части для хранения схем.
4. Добавление возможности экспорта.
5. Проведение тестирования, подготовка пояснительной записи.

Тема 7: Система визуального программирования (аналог Scratch) **Описание:** Образовательная платформа, позволяющая создавать алгоритмы с помощью визуальных блоков. Подходит для начинающих студентов, чтобы освоить базовые логические конструкции.

Техническое задание:

- **Серверная часть (Python, FastAPI):**
 - Хранение проектов пользователей.
 - API для сохранения и загрузки программ.

- **Клиентская часть (JavaScript):**
 - Интерфейс визуального программирования (движение, условия, циклы).
 - Область вывода и симуляции работы алгоритма.

План выполнения:

1. Разработка базовой библиотеки визуальных блоков.
2. Интеграция исполнителя для запуска алгоритма.
3. Настройка взаимодействия с сервером.
4. Реализация пользовательского кабинета.
5. Подготовка презентации и пояснительной записи.

Тема 8: Интерактивный тренажёр по логическим выражениям

Описание: Приложение для практики составления логических выражений, построения таблиц истинности и преобразования логических формул.

Техническое задание:

- **Серверная часть (Python, FastAPI):**
 - Генерация задач по логике.
 - Проверка правильности выражений.
 - Сохранение результатов и прогресса пользователя.
- **Клиентская часть (JavaScript):**
 - Ввод выражений, визуализация таблицы истинности.
 - Подсказки и анализ ошибок.

План выполнения:

1. Разработка механизма разбора логических выражений.
2. Создание визуального представления таблиц истинности.
3. Интеграция системы подсказок и проверки.
4. Сборка интерфейса и серверной части.
5. Финальное тестирование и отчётность.

Тема 9: Платформа визуализации рекурсии **Описание:** Система для обучения рекурсии, демонстрирующая процесс вызова функций в виде дерева рекурсий с визуальной анимацией и комментариями.

Техническое задание:

- **Серверная часть (Python, FastAPI):**
 - Приём пользовательских рекурсивных функций.
 - Генерация дерева вызовов и возвращаемых значений.
- **Клиентская часть (JavaScript):**
 - Анимация рекурсивного дерева.
 - Контроль шагов, пауза/воспроизведение.

План выполнения:

1. Реализация трекера вызовов и стеков.
2. Разработка визуализации дерева вызовов.
3. Интеграция с редактором кода и выводом.
4. Тестирование с типовыми задачами (факториал, фибоначчи).

5. Подготовка документации и презентации.

Тема 10: Система задач по циклам и условиям с автопроверкой

Описание: Практический тренажер для студентов, в котором можно решать задачи на циклы и условия. Проверка решений выполняется автоматически на сервере с различными входными данными.

Техническое задание:

- **Серверная часть (Python, FastAPI):**
 - База задач и тестов.
 - Автоматическая проверка решений.
 - Хранение прогресса и результатов студентов.
- **Клиентская часть (JavaScript):**
 - Редактор кода с возможностью запуска.
 - Вывод результата и замечаний.

План выполнения:

1. Формирование базы задач и тестов.
2. Настройка системы проверки кода.
3. Разработка интерфейса решения задач.
4. Тестирование системы и подведение итогов.
5. Подготовка к защите.

Тема 11: Конструктор алгоритмов с псевдокодом

Описание: Веб-приложение, в котором пользователь создает алгоритмы в формате псевдокода с визуальным редактором, а система автоматически преобразует их в Python-код.

Техническое задание:

- **Серверная часть (Python, FastAPI):**
 - Хранение сессий, проектов и версий алгоритмов.
 - API для трансляции псевдокода в Python.
- **Клиентская часть (JavaScript):**
 - Интерфейс создания псевдокода блоками.
 - Просмотр сгенерированного Python-кода в реальном времени.

План выполнения:

1. Разработка визуального редактора псевдокода.
2. Реализация логики преобразования в Python.
3. Связь клиентской и серверной части.
4. Реализация хранения и загрузки проектов.
5. Подготовка демонстрации и документации.

Тема 12: Тренажёр логических выражений и таблиц истинности

Описание: Образовательное приложение для изучения логических операций, построения таблиц истинности и упрощения логических формул.

Техническое задание:

- **Серверная часть (Python, FastAPI):**

- Генерация задачий по логике.
- Проверка правильности решений.
- Клиентская часть (*JavaScript*):
 - Интерфейс ввода логических выражений.
 - Автоматическое построение таблицы истинности.

План выполнения:

1. Реализация парсера логических выражений.
2. Создание таблицы истинности в клиентской части.
3. Организация обмена данными с сервером.
4. Создание базы задачий.
5. Подготовка пояснительной записи.

Тема 13: Визуализация выполнения рекурсивных функций **Описание:** Сервис, отображающий выполнение рекурсивной функции в виде дерева вызовов с анимацией и пояснениями на каждом этапе.

Техническое задание:

- Серверная часть (*Python, FastAPI*):
 - Обработка пользовательского кода.
 - Генерация данных для визуализации.
- Клиентская часть (*JavaScript*):
 - Анимация дерева вызовов.
 - Отображение стеков и возвращаемых значений.

План выполнения:

1. Анализ типовых рекурсивных задач.
2. Построение модели вызовов и возвратов.
3. Создание анимации на клиенте.
4. Интеграция и тестирование.
5. Сдача проекта с демонстрацией.

Тема 14: Генератор лабиринтов и поиск пути (BFS/DFS/A*)

Описание: Веб-приложение для генерации лабиринтов и пошагового отображения алгоритмов поиска пути с визуализацией (BFS, DFS, A*).

Техническое задание:

- Серверная часть (*Python, FastAPI*):
 - Генерация лабиринтов и расчет маршрутов.
- Клиентская часть (*JavaScript*):
 - Отображение лабиринта и визуализация алгоритма поиска.

План выполнения:

1. Разработка генератора случайных лабиринтов.
2. Реализация поиска пути алгоритмами BFS/DFS/A*.
3. Создание визуализации на клиенте.
4. Интеграция всех компонентов.
5. Подготовка к защите.

Тема 15: Визуализатор алгоритма Евклида (поиск НОД) **Описание:** Образовательный проект для пошагового отображения работы алгоритма Евклида по нахождению наибольшего общего делителя двух чисел.

Техническое задание:

- **Серверная часть (Python, FastAPI):**
 - Реализация алгоритма Евклида.
 - Генерация шагов выполнения.
- **Клиентская часть (JavaScript):**
 - Графическое представление делений.
 - Комментарии к каждому шагу.

План выполнения:

1. Описание логики алгоритма Евклида.
2. Реализация API для пошагового ответа.
3. Визуализация на клиенте.
4. Добавление интерактива и обучения.
5. Подготовка презентации и пояснений.

Тема 16: Система визуального сравнения алгоритмов сортировки **Описание:** Образовательный веб-инструмент, позволяющий сравнивать эффективность различных алгоритмов сортировки на одинаковых наборах данных. Пользователь может выбрать алгоритмы, задать размер массива и наблюдать разницу в производительности и скорости.

Техническое задание:

- **Серверная часть (Python, FastAPI):**
 - Реализация алгоритмов сортировки (Bubble, Insertion, Merge, Quick).
 - Генерация случайных массивов и логирование шагов.
- **Клиентская часть (JavaScript):**
 - Анимация процесса сортировки.
 - Сравнение времени выполнения и количества операций.

План выполнения:

1. Имплементация серверных функций сортировки с логированием шагов.
2. Создание UI с выбором алгоритмов и параметров сортировки.
3. Реализация анимаций и таймеров.
4. Тестирование системы и анализ производительности.
5. Подготовка проекта и отчета.

Тема 17: Конструктор алгоритмов поиска в массиве и строке **Описание:** Приложение для визуального и практического изучения алгоритмов линейного и бинарного поиска, поиска подстрок и шаблонов в строках.

Техническое задание:

- **Серверная часть (Python, FastAPI):**
 - Реализация алгоритмов поиска.
 - API для получения шагов выполнения поиска.

- **Клиентская часть (JavaScript):**
 - Отображение массива или строки.
 - Анимация каждого шага поиска.

План выполнения:

1. Создание базы поисковых задач.
2. Имплементация API шагов поиска.
3. Визуализация поиска на клиента.
4. Добавление обучающих комментариев.
5. Презентация и тестирование.

Тема 18: Система проверки задач на работу со списками и строками

Описание: Практическая онлайн-платформа для закрепления навыков работы со списками и строками на Python. Система автоматически проверяет решения и выдает рекомендации.

Техническое задание:

- **Серверная часть (Python, FastAPI):**
 - База задач с уровнями сложности.
 - Автопроверка решений по тест-кейсам.
- **Клиентская часть (JavaScript):**
 - Редактор кода с тестированием.
 - Таблица результатов и подсказки.

План выполнения:

1. Формирование каталога задач.
2. Настройка среды автопроверки.
3. Разработка пользовательского интерфейса.
4. Подключение базы данных для учёта прогресса.
5. Подготовка финального релиза.

Тема 19: Генератор задач по арифметике и логике с автооценкой

Описание: Сервис для генерации простых задач по арифметике, условиям и логике с автоматической проверкой и сохранением результатов.

Техническое задание:

- **Серверная часть (Python, FastAPI):**
 - Генерация случайных задач (арифметика, логика).
 - Проверка ответов и выдача баллов.
- **Клиентская часть (JavaScript):**
 - Отображение задач и формы для ответов.
 - Подсказки, таймер и статистика.

План выполнения:

1. Создание генератора арифметических и логических заданий.
2. Имплементация системы оценки и хранения баллов.
3. Визуальное оформление страницы тренировки.
4. Подготовка пользовательской статистики.
5. Завершение и подготовка защиты.

Тема 20: Платформа модульного обучения по темам ОАиП *Описание:* Онлайн-система, где учебный курс разбит на модули (переменные, операторы, условия, циклы, функции и т.д.). После каждого модуля — задания и автоматическая проверка.

Техническое задание:

- Серверная часть (*Python, FastAPI*):
 - База учебных модулей и заданий.
 - Авторизация и отслеживание прогресса.
- Клиентская часть (*JavaScript*):
 - Интерфейс навигации по модулям.
 - Встроенный редактор кода и проверка решений.

План выполнения:

1. Структурирование модулей по темам.
2. Создание системы хранения и выдачи заданий.
3. Реализация проверочной логики и отслеживания.
4. UI/UX интерфейс ученика.
5. Презентация и защита проекта.

Тема 21: Интерактивный отладчик псевдокода *Описание:* Образовательное приложение, позволяющее пользователю писать псевдокод, запускать его по шагам и отслеживать состояние переменных. Подходит для начального уровня изучения логики алгоритмов.

Техническое задание:

- Серверная часть (*Python, FastAPI*):
 - Интерпретация псевдокода.
 - Обработка и хранение состояния исполнения.
- Клиентская часть (*JavaScript*):
 - Поле для ввода псевдокода.
 - Кнопка «шаг за шагом», отображение переменных и стеков вызова.

План выполнения:

1. Разработка синтаксиса и парсера псевдокода.
2. Реализация логики пошагового исполнения.
3. Отображение переменных и операций в интерфейсе.
4. Связь клиента с API отладки.
5. Подготовка презентации.

Тема 22: Визуальный редактор блоков кода (Code Block Designer) *Описание:* Веб-приложение, в котором пользователь может создавать блоки кода (условия, циклы, функции) с помощью визуального редактора и конвертировать их в Python-скрипт.

Техническое задание:

- Серверная часть (*Python, FastAPI*):

- Обработка логики конвертации блоков в код.
- Хранение пользовательских скриптов.
- **Клиентская часть (JavaScript):**
 - Drag-and-drop блоки с параметрами.
 - Просмотр и редактирование итогового Python-кода.

План выполнения:

1. Создание визуальных блоков и их связей.
2. Настройка правил генерации Python-кода.
3. Связь редактора с API сервера.
4. Добавление сохранения и загрузки проектов.
5. Финальное тестирование.

Тема 23: Тренажёр по трассировке кода **Описание:** Система для пошагового выполнения готовых алгоритмов с возможностью остановки, просмотра текущих переменных и комментариев к каждой строке. Полезно для освоения логики алгоритмов и работы с отладчиком.

Техническое задание:

- **Серверная часть (Python, FastAPI):**
 - Хранение заданий и состояния исполнения.
 - Поддержка контрольных точек (breakpoints).
- **Клиентская часть (JavaScript):**
 - Поле с подсветкой кода.
 - Кнопки «шаг вперёд», «обратно», просмотр переменных.

План выполнения:

1. Подбор и описание обучающих алгоритмов.
2. Реализация движка для трассировки кода.
3. Визуализация состояния исполнения.
4. Интеграция с базой заданий.
5. Защита проекта.

Тема 24: Платформа задач по графам и деревьям **Описание:** Образовательный сервис с задачами на графы и деревья. Пользователь должен реализовать алгоритмы (DFS, BFS, поиск путей, минимальное остовное дерево), а система проверяет корректность и визуализирует результат.

Техническое задание:

- **Серверная часть (Python, FastAPI):**
 - Автоматическая проверка решений.
 - Генерация тестовых графов и деревьев.
- **Клиентская часть (JavaScript):**
 - Редактор кода и визуализация структуры графа.
 - Подсветка маршрутов, комментарии к шагам.

План выполнения:

1. Формирование набора задач по графикам.

2. Реализация генератора и визуализатора.
3. Разработка интерфейса отправки решений.
4. Интеграция проверок и обратной связи.
5. Подготовка итоговой документации.

Тема 25: Конструктор и симулятор работы стеков и очередей **Описание:** Веб-инструмент для визуального изучения структур данных «стек» и «очередь» с возможностью ручного добавления и удаления элементов, пошагового просмотра операций и отображения внутреннего состояния структуры.

Техническое задание:

- Серверная часть (*Python, FastAPI*):
 - Хранение пользовательских сессий и истории операций.
 - Проверка корректности выполняемых операций.
- Клиентская часть (*JavaScript*):
 - Визуальный редактор для работы со структурами.
 - Подсказки, отладка, комментарии.

План выполнения:

1. Реализация логики работы стека и очереди.
2. Разработка визуальных элементов интерфейса.
3. Сценарии тестирования операций (*push, pop, enqueue, dequeue*).
4. Интеграция всех компонентов.
5. Презентация готового решения.

Тема 26: Онлайн-симулятор выполнения кода (Code Runner Sandbox)

Описание: Веб-приложение, позволяющее пользователю вводить и запускать простой Python-код с ограничением по времени и ресурсам. Подходит для безопасного тестирования фрагментов алгоритмов.

Техническое задание:

- Серверная часть (*Python, FastAPI*):
 - Контейнеризация и запуск кода с ограничениями.
 - Логирование ошибок и результатов исполнения.
- Клиентская часть (*JavaScript*):
 - Интерфейс для ввода кода и вывода результатов.
 - Панель ошибок и таймер выполнения.

План выполнения:

1. Настройка безопасной среды запуска Python-кода.
2. Разработка интерфейса с редактором и консолью.
3. Обработка ошибок и вывод результата.
4. Ведение истории запусков.
5. Финальная сборка и защита.

Тема 27: Визуализатор стека вызовов (Call Stack Visualizer) **Описание:** Приложение, демонстрирующее, как работает стек вызовов в Python при

последовательных и рекурсивных вызовах функций. Полезно для понимания концепций вложенностии и возвратов.

Техническое задание:

- **Серверная часть (Python, FastAPI):**
 - Анализ пользовательского кода.
 - Отправка данных о каждом вызове и возврате.
- **Клиентская часть (JavaScript):**
 - Визуализация стека вызовов в реальном времени.
 - Пошаговый режим исполнения.

План выполнения:

1. Разработка парсера кода и трекера вызовов.
2. Создание визуального компонента стека.
3. Реализация взаимодействия с пользователем.
4. Проведение тестирования с примерами.
5. Подготовка проекта к защите.

Тема 28: Генератор учебных задач с автоматическим контролем решений **Описание:** Приложение, в котором преподаватель может задавать условия задач, а система автоматически генерирует варианты входных данных и проверяет решения студентов по встроенным шаблонам.

Техническое задание:

- **Серверная часть (Python, FastAPI):**
 - Интерфейс преподавателя для создания задач.
 - Проверка решений с помощью эталонного алгоритма.
- **Клиентская часть (JavaScript):**
 - Интерфейс студента для ввода решений.
 - Проверка, результаты и обратная связь.

План выполнения:

1. Проектирование структуры задач и шаблонов.
2. Разработка панели преподавателя.
3. Настройка автопроверки решений.
4. Разработка интерфейса студента.
5. Проведение тестирования и защита.

Тема 29: Онлайн-платформа задач с выбором алгоритма **Описание:** Пользователь получает описание задачи и должен выбрать подходящий алгоритм из списка (жадный, динамическое программирование, жадный, рекурсивный и т.д.), реализовать его и пройти проверку.

Техническое задание:

- **Серверная часть (Python, FastAPI):**
 - Каталог задач и типов алгоритмов.
 - Автоматическая проверка решений.
- **Клиентская часть (JavaScript):**
 - Навигация по задачам, редактор кода, проверка результатов.

План выполнения:

1. Создание базы задач с пометкой применимого алгоритма.
2. Разработка проверки решений.
3. Интерфейс для тренировки и обучения.
4. Система наград и рейтингов.
5. Финализация и защита.

Тема 30: Система цифрового зачётного ведомости (Academic Gradebook)

Описание: Веб-приложение для преподавателей, позволяющее вести учет оценок студентов, проверку выполнения задач и генерацию отчетов по курсу.

Техническое задание:

- Серверная часть (*Python, FastAPI*):
 - Авторизация преподавателей и студентов.
 - Работа с заданиями, оценками и результатами.
- Клиентская часть (*JavaScript*):
 - Панель преподавателя с формами и таблицами.
 - Просмотр успеваемости студентами.

План выполнения:

1. Проектирование структуры базы (пользователи, задания, оценки).
2. Реализация интерфейса управления ведомостью.
3. Генерация отчётности.
4. Проведение тестирования.
5. Подготовка финального доклада.