**Профессиональное образовательное частное учреждение**

**«Московский кооперативный техникум им.Г.Н.Альтшуля»**

ДНЕВНИК-ОТЧЕТ

по практической подготовке

**(**практика по профилю специальности**)**

# Студента(-ки) Харитонова Николая

Владимировича

(*фамилия, имя, отчество)*

# Группа ИС-41

# Специальность Информационные системы и программирование

Срок сдачи дневника – отчета « » 20 г. Дата проведения конференции « » 20 г.

2024-2025 учебный год

1. **ОБЩИЕ СВЕДЕНИЯ**

База практической подготовки (профильной организации)

«ООО ПИКВИТА»

Период с\_\_\_\_\_\_\_\_01.11\_\_\_\_\_\_\_\_\_\_\_по\_\_\_\_\_\_\_\_\_\_\_\_14.12\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_24\_г.

Руководитель от профильной организации (Ф.И.О., должность, образование, стаж работы)

ФИО - Зайцев Алексей Анатольевич

Должность – Генеральный директор

Образование – Высшее, бакалавриат, магистратура

Стаж – 40 лет

Результаты освоения практической подготовки

Руководитель от техникума

« » 20 г

## КАЛЕНДАРНО-ТЕМАТИЧЕСКИЙ ПЛАН

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ п/п** | Наименование темы и содержание работы | Количество дней | | Отметка о выполнении |
| по  плану | по  факту |
| 1. | Вводный инструктаж. Характеристика предприятия | 1 | 1 | Выполнено |
| 2. | Основные виды и процедуры обработки информации, модели и методы решения задач обработки информации. | 1 | 1 | Выполнено |
| 3. | Основные платформы для создания, исполнения и управления информационной системой. | 1 | 1 | Выполнено |
| 4. | Основные модели построения информационных систем, их структуру, особенности и области применения. | 1 | 1 | Выполнено |
| 5. | Платформы для создания, исполнения и управления информационной системой | 2 | 2 | Выполнено |
| 6. | Основные процессы управления проектом разработки. | 2 | 2 | Выполнено |
| 7. | Методы и средства проектирования, разработки и тестирования информационных систем. | 1 | 1 | Выполнено |
| 8. | Национальная и международная система стандартизации и сертификации и система обеспечения качества продукции, методы контроля качества. | 2 | 2 | Выполнено |
| 9. | Сервисно - ориентированные архитектуры. | 2 | 2 | Выполнено |
| 10. | Важность рассмотрения всех возможных вариантов и получения наилучшего решения на основе анализа и интересов клиента. | 2 | 2 | Выполнено |
| 11. | Основные понятия системного анализа. | 1 | 1 | Выполнено |
| 12. | Реинжиниринг бизнес-процессов | 2 | 2 | Выполнено |
| 13. | Система обеспечения качества продукции. | 1 | 1 | Выполнено |
| 14. | Методы контроля качества в соответствии со стандартами | 1 | 1 | Выполнено |
| 15. | Объектно-ориентированное программирование. | 2 | 2 | Выполнено |
| 16. | Спецификации языка программирования, принципы создания графического пользовательского интерфейса (GUI). | 2 | 2 | Выполнено |
| 17. | Создание файлового ввода-вывода. | 1 | 1 | Выполнено |
| 18. | Создания сетевого сервера и сетевого клиента. | 2 | 2 | Выполнено |
| 19. | Особенности и области применения. | 1 | 1 | Выполнено |
| 20. | Особенности программных средств, используемых в разработке ИС | 1 | 1 | Выполнено |
| 21. | Оформление отчета. | 1 | 1 | Выполнено |
| **ИТОГО:** | | 30 | 30 |  |

1. **ДНЕВНИК-ОТЧЕТ ПО ВЫПОЛНЕННОЙ РАБОТЕ**

|  |  |  |
| --- | --- | --- |
| **Дата** | **№ темы Название темы**  **Содержание работы** | **Оценка и подпись**  **руководителя** |
| 01.11 | Вводный инструктаж на предприятии, занимающемся информационными системами и программированием, играет ключевую роль в адаптации новых сотрудников и знакомстве их с корпоративной культурой, рабочими процессами, а также используемыми технологиями и системами. Ниже приведен примерный план такого инструктажа.  1. Приветствие и представление  Приветствие: Краткое обращение руководителя или HR-специалиста к новому сотруднику. Представление команды: Краткая информация о каждом члене команды, его роли и функциях.  2. Общая информация о компании  История компании: Краткий обзор истории создания и развития компании. Миссия и ценности: Основные цели компании и ценности, которые она поддерживает. Структура компании: Описание организационной структуры, отделов и их функций.  3. Корпоративная культура  Нормы и правила: Описание dress code, рабочего времени, поведения на рабочем месте. Командная работа: Основные принципы взаимодействия внутри команды и с другими отделами. Обратная связь: Как и когда сотрудники могут получать и предоставлять обратную связь.  4. Основные направления работы  Информационные системы: Обзор используемых информационных систем, их назначения и роли в бизнесе. Программирование: Основные языки программирования и технологии, которые применяются в компании (например, Python, Java, C#, базы данных, фреймворки и др.).  5. Процессы разработки и управления проектами  Методологии разработки: Обзор Agile, Scrum, Kanban и других подходов, используемых в компании. Инструменты управления проектами: Знакомство с используемыми системами (например, Jira, Trello, Asana). Работа с документацией: Как оформляются проекты, технические требования, документация.  6. Безопасность и защита данных  Политики безопасности: Основные правила по обеспечению информационной безопасности и защиты данных. Обучение по безопасности: Обязанности сотрудников по соблюдению правил безопасности.  7. Техническое обеспечение  Оборудование и ПО: Информация о предоставляемом оборудовании, программном обеспечении и доступе к ресурсам. Поддержка и обслуживание: Процесс обращения в IT-отдел при возникновении технических проблем.  8. Обучение и развитие  Программы обучения: Обзор доступных внутренних и внешних курсов, семинаров. Карьера и рост: Возможности карьерного роста внутри компании.  9. Вопросы и ответы  Обсуждение: Выделение времени для ответа на вопросы нового сотрудника и обсуждения возможных неясностей.  10. Завершение инструктажа  Подведение итогов: Краткое резюме основных моментов инструктажа. Настройка на работу: Мотивационное напутствие и пожелание успехов на новом месте.  Заключение  Такой вводный инструктаж позволяет новым сотрудникам быстрее адаптироваться к рабочей среде, понять внутренние процессы и установленные практики. Наличие четкой структуры инструктажа способствует созданию комфортной и продуктивной атмосферы, что в свою очередь положительно сказывается на работе всего предприятия. |  |
| 02.11  03.11  04.11 | ВЫХОДНОЙ ДЕНЬ  ВЫХОДНОЙ ДЕНЬ  ВЫХОДНОЙ ДЕНЬ |  |

|  |  |  |
| --- | --- | --- |
| 05.11 | Обработка информации является ключевым аспектом работы с данными в современном мире. Существует множество видов обработки информации, а также модели и методы, используемые для решения задач, связанных с обработкой информации. Рассмотрим основные из них.  Основные виды обработки информации  Сбор данных:  Определение: Первичный этап, где происходит сбор необходимой информации. Методы: Опросы, анкетирование, автоматизированные системы мониторинга.  Хранение данных:  Определение: Сохранение собранной информации для дальнейшего использования. Методы: Реляционные базы данных (SQL), NoSQL базы данных, облачные хранилища.  Обработка данных:  Определение: Преобразование собранной информации в полезный формат для анализа. Методы: Фильтрация, агрегация, сортировка, преобразование.  Анализ данных:  Определение: Извлечение инсайтов и значений из обработанной информации. Методы: Статистический анализ, машинное обучение, визуализация или построение отчетов.  Передача информации:  Определение: Распространение информации среди заинтересованных сторон. Методы: Электронная почта, мессенджеры, системы совместной работы (например, Slack, Microsoft Teams).  Уничтожение данных:  Определение: Порядок удаления устаревшей или ненужной информации. Методы: Шифрование перед уничтожением, физическое разрушение носителей.  Процедуры обработки информации  Идентификация проблемы:  Выявление нужд и требований пользователей.  Сбор и подготовка данных:  Сбор данных из различных источников и их подготовка к анализу.  Обработка и анализ:  Применение различных методов для извлечения полезной информации из данных.  Интерпретация результатов:  Выводы и рекомендации на основе полученных результатов анализа.  Документация:  Создание отчетов и документации для заинтересованных сторон.  Внедрение решений:  Реализация рекомендаций и выводов в практическую деятельность.  Модели обработки информации  Модель ввода-вывода:  Основана на концепции ввода данных, их обработки и вывода результатов.  Модель базы данных:  Структурирование данных в виде таблиц для удобства обработки и поиска (реляционные, документные, графовые базы данных).  Статистическая модель:  Используется для анализа данных (например, линейная регрессия, методы кластеризации).  Модели на основе правил и логики:  Используются в системах, основанных на правилах, и для обработки экспертных знаний.  Машинное обучение и искусственный интеллект:  Используются для решения сложных задач обработки информации, таких как классификация и прогнозирование.  Методы решения задач обработки информации  Статистические методы:  Регрессионный анализ, дисперсионный анализ, корреляция и др.  Алгоритмы машинного обучения:  Классификация, кластеризация, регрессия, нейронные сети.  Методы оптимизации:  Линейное программирование, жадные алгоритмы, методы выдачи.  Визуализация данных:  Использование графиков, диаграмм, интерактивных панелей для представления данных.  Программирование и создание программного обеспечения:  Разработка специализированных приложений для обработки и анализа данных.  Системы поддержки принятия решений (DSS):  Инструменты, помогающие пользователям принимать обоснованные решения на основе анализа данных.  Каждая из этих областей имеет свои специфические подходы и инструменты. Выбор конкретного метода обработки зависит от задачи, характера данных и требований к конечному результату. |  |

## 

|  |  |  |
| --- | --- | --- |
| 06.11 | Обработка информации – это сложный и многоступенчатый процесс, который включает в себя различные виды и процедуры работы с данными, а также применение моделей и методов для решения поставленных задач. В этом ответе мы рассмотрим основные виды и процедуры обработки информации, а также используемые методы и модели.  Основные виды обработки информации  Сбор данных:  Определение: Процесс получения данных из различных источников. Методы: Опросы, интервью, автоматизированные системы сбора данных, веб-скрейпинг и сенсоры.  Хранение данных:  Определение: Сохранение данных для будущего доступа и использования. Методы: Реляционные базы данных (например, MySQL, PostgreSQL), NoSQL базы данных (например, MongoDB), облачные хранилища (например, Google Drive, Amazon S3).  Обработка данных:  Определение: Преобразование, очистка и организация данных для дальнейшего анализа. Методы: ETL-процессы (извлечение, преобразование, загрузка), нормализация, агрегация, фильтрация, сортировка.  Анализ данных:  Определение: Исследование и выявление закономерностей или аномалий в данных. Методы: Статистический анализ, машинное обучение, Data Mining, применение алгоритмов кластеризации, регрессии, классификации.  Визуализация данных:  Определение: Презентация данных в графическом виде для лучшего понимания результатов анализа. Методы: Использование графиков, диаграмм, карт, инструментов визуализации (например, Tableau, Power BI).  Передача информации:  Определение: Обмен информацией между сторонами. Методы: Отчеты, дашборды, внутренние и внешние коммуникационные каналы.  Уничтожение данных:  Определение: Процесс безопасного удаления данных, которые больше не нужны. Методы: Шифрование и физическое уничтожение носителей данных.  Процедуры обработки информации  Идентификация проблемы:  Определение проблем, которые необходимо решить на основе наличной информации.  Сбор информации:  Сбор данных из разных источников, которые могут помочь в анализе.  Проверка качества данных:  Оценка собранных данных на предмет их полноты, актуальности и достоверности.  Обработка данных:  Применение алгоритмов и методов для обработки и анализа собранных данных.  Анализ данных:  Использование статистических и аналитических методов для выявления тенденций и закономерностей.  Интерпретация результатов:  Осмысление и представление результатов анализа с целью сделать обоснованные выводы.  Документирование и представление:  Подготовка отчетов и документирование процесса обработки; визуализация результатов анализа.  Модели обработки информации  Модель обработки данных:  Описывает как данные собираются, хранятся, обрабатываются и передаются, включая различные методы и инструменты.  Модель систем обработки информации:  Определяет роли различных компонентов системы, таких как базы данных, программное обеспечение и оборудования.  Модель анализа данных:  Включает в себя различные подходы к анализу данных, такие как статистические методы, методы машинного обучения и искусственного интеллекта.  Методы решения задач обработки информации  Статистические методы:  Приемы и техники анализа данных, включающие описательную статистику, предполагаемую статистику и методы проверки гипотез.  Алгоритмы машинного обучения:  Методы, позволяющие моделям учиться на данных и делать предсказания или классификации (например, линейная регрессия, дерево решений, нейронные сети).  Методы Data Mining:  Техники, направленные на обнаружение закономерностей в больших объемах данных, такие как кластеризация и ассоциация.  Методы оптимизации:  Применение математических алгоритмов для нахождения наилучших решений в различных ситуациях (например, линейное программирование, генетические алгоритмы).  Интуитивные и эвристические методы:  Практические подходы к решению задач, которые основываются на эмпирическом опыте и могут помочь найти решения, когда традиционные методы неэффективны.  Таким образом, обработка информации включает в себя множество аспектов, начиная от сбора и хранения данных до их анализа и визуализации. Разнообразие методов и подходов позволяет выбирать наиболее подходящие решения для конкретных задач в зависимости от целей и доступных ресурсов. |  |
| 07.11 | Создание, исполнение и управление информационными системами требует использования различных платформ и инструментов, которые поддерживают нужные функции на каждом этапе жизненного цикла системы. Рассмотрим основные типы платформ, используемых в этом контексте.  1. Платформы разработки  Системы управления базами данных (СУБД):  Примеры: MySQL, PostgreSQL, Microsoft SQL Server, Oracle Database, MongoDB.  Описание: Служат для хранения и управления данными, обеспечивая возможность выполнения запросов и транзакций.  Системы разработки приложений:  Примеры: Microsoft Visual Studio, Eclipse, JetBrains IntelliJ IDEA.  Описание: Инструменты для разработки программного обеспечения, позволяющие писать код, отлаживать и тестировать приложения.  Платформы для создания веб-приложений:  Примеры: Ruby on Rails, Django, Flask, ASP.NET, Laravel.  Описание: Фреймворки, которые упрощают разработку веб-приложений, предлагая готовые решения и библиотеки.  2. Платформы исполнения  Облачные платформы:  Примеры: Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP).  Описание: Предлагают инфраструктуру как услугу (IaaS), платформу как услугу (PaaS) и программное обеспечение как услугу (SaaS), позволяя развертывать приложения в облаке.  Платформы контейнеризации:  Примеры: Docker, Kubernetes.  Описание: Позволяют упаковывать приложения и их зависимости в контейнеры для обеспечения переносимости и удобного управления.  3. Платформы управления  Системы управления проектами:  Примеры: JIRA, Trello, Asana, Microsoft Project.  Описание: Помогают управлять задачами, отслеживать прогресс и планировать работу команд.  Системы мониторинга и анализа:  Примеры: Prometheus, Grafana, ELK Stack (Elasticsearch, Logstash, Kibana).  Описание: Позволяют отслеживать производительность и доступность информационных систем, а также анализировать журналы и метрики.  Платформы для DevOps:  Примеры: Jenkins, GitLab CI/CD, CircleCI.  Описание: Инструменты, которые помогают интегрировать процесс разработки и эксплуатации, автоматизируя процессы развертывания и тестирования.  4. Платформы безопасности  Системы управления безопасностью:  Примеры: Splunk, IBM Security QRadar.  Описание: Решения для обнаружения и реагирования на угрозы, а также для управления инцидентами безопасности.  Инструменты аутентификации и управления доступом:  Примеры: Auth0, Okta, AWS IAM.  Описание: Платформы, которые встраивают механизмы управления идентификацией и доступом к приложениям и данным.  Заключение  Чтобы успешно разрабатывать, развертывать и управлять информационной системой, важно выбрать правильные платформы на каждом этапе разработки. Выбор инструментов зависит от специфических требований проекта, доступных ресурсов и других факторов. Однако интеграция различных платформ и инструментов позволяет построить эффективную и устойчивую информационную систему. |  |
| 08.11 | Построение информационных систем (ИС) может быть основано на различных моделях, каждая из которых имеет свои особенности, преимущества и области применения. Рассмотрим основные модели, используемые в разработке и внедрении информационных систем.  1. Модель «Клиент-сервер»  Структура:  Клиент: Предоставляет интерфейс для пользователей, отправляет запросы на сервер.  Сервер: Обрабатывает запросы клиентов, выполняет бизнес-логику и управляет доступом к данным.  Особенности:  Клиенты могут быть разными: веб-приложения, настольные программы и мобильные приложения.  Сервер может быть распределённым, чтобы обеспечить высокую доступность и балансировку нагрузки.  Позволяет многопользовательский доступ к данным.  Области применения:  Веб-сайты и интернет-приложения.  Системы управления базами данных.  Облачные услуги и SaaS.  2. Модель «Микросервисы»  Структура:  Построение системы из множества малых, независимых сервисов.  Каждый микросервис отвечает за конкретную функциональность и может быть независимо развернут.  Особенности:  Лёгкость разработки и развертывания отдельных компонентов.  Возможность использования различных технологий для каждого микросервиса.  Упрощает масштабирование и обновление системы.  Области применения:  Современные веб и мобильные приложения.  Сложные системы с высокой нагрузкой (например, платформы e-commerce).  Приложения, требующие частых обновлений и развертываний.  3. Модель «Слойная архитектура»  Структура:  Обычно включает три основных слоя:  Презентационный слой (интерфейс пользователя).  Прикладной слой (бизнес-логика).  Слой данных (доступ к данным).  Особенности:  Чёткое разделение обязанностей между слоями.  Упрощает поддержку и модификацию системы.  Возможно повторное использование слоёв в других проектах.  Области применения:  Традиционные корпоративные приложения.  Системы, где важна структура и модульность (например, ERP).  4. Модель «Событийно-ориентированная архитектура»  Структура:  Система реагирует на события, возникающие в результате взаимодействия с пользователями или другими системами.  Обычно включает продюсеров событий, брокеров и потребителей событий.  Особенности:  Высокая степень асинхронности.  Легкость интеграции различных систем и сервисов.  Упрощает анализ и реагирование на бизнес-события.  Области применения:  Реализация систем бизнес-аналитики.  Приложения, работающие в реальном времени (например, потоковая обработка данных).  Системы с высокой уровнем взаимодействия между компонентами.  5. Модель «Программное обеспечение как услуга» (SaaS)  Структура:  Приложения развернуты и управляются на облаке, пользователи имеют доступ через интернет.  Все клиенты используют одну и ту же версию приложения, с изолированными данными.  Особенности:  Отсутствие необходимости в локальной установке и обслуживании со стороны пользователей.  Подписочная модель оплаты.  Автоматические обновления и управление безопасностью со стороны провайдера.  Области применения:  CRM-системы (например, Salesforce).  Системы управления проектами (например, Asana, Trello).  Облачные офисные приложения (например, Google Workspace).  Заключение  Каждая из вышеописанных моделей имеет свои сильные и слабые стороны, и выбор конкретной модели зависит от требований проекта, масштаба, бюджета и других факторов. Правильный выбор модели позволяет повысить эффективность разработки, упростить поддержку и обеспечить масштабируемость системы в будущем. |  |
| 09.11  10.11 | ВЫХОДНОЙ ДЕНЬ  ВЫХОДНОЙДЕНЬ |  |
| 11.11 | Создание, исполнение и управление информационными системами (ИС) требует использование различных платформ и инструментов, которые могут помочь в разработке, развертывании и поддержке этих систем. Рассмотрим ключевые категории платформ, используемых для этих целей.  1. Платформы для разработки  1.1. Инструменты для программирования  IDE (интегрированные среды разработки):  Visual Studio: популярная среда для разработки на языках C#, .NET и других.  Eclipse: универсальная IDE для разработки на Java и других языках.  PyCharm: оптимизированная для Python IDE с множеством инструментов для анализа кода.  1.2. Фреймворки  Web Frameworks:  Django: высокоуровневый фреймворк для разработки веб-приложений на Python.  Ruby on Rails: фреймворк для создания веб-приложений на Ruby.  Spring: мощный фреймворк для Java с поддержкой микросервисов.  Mobile Frameworks:  React Native: позволяет создавать нативные мобильные приложения с использованием JavaScript и React.  Flutter: инструмент от Google для создания кроссплатформенных приложений на Dart.  2. Платформы для исполнения  2.1. Операционные системы и серверы  Windows Server: популярная ОС для хостинга корпоративных приложений и веб-сайтов.  Linux: широкий выбор дистрибутивов (например, Ubuntu, CentOS), часто используемых для серверных приложений.  2.2. Виртуализация и контейнеризация  Docker: платформа для контейнеризации приложений, позволяющая легко развертывать и масштабировать ИС.  Kubernetes: система для управления контейнеризированными приложениями, обеспечивающая автоматизацию развертывания, масштабирования и управления контейнерами.  3. Платформы для управления  3.1. Базы данных  Реляционные базы данных:  MySQL: популярная СУБД с открытым исходным кодом.  PostgreSQL: мощная реляционная СУБД с поддержкой сложных запросов и транзакций.  NoSQL базы данных:  MongoDB: ориентированная на документы база данных, обеспечивающая гибкость при работе с большими объемами данных.  Cassandra: распределенная NoSQL база данных, подходящая для обработки огромных объемов данных.  3.2. Инструменты управления проектами и DevOps  JIRA: система для управления проектами и отслеживания задач, популярная в Agile-методологиях.  GitLab и GitHub: платформы для управления версиями кода и коллаборации, предоставляют инструменты CI/CD для автоматизации развертывания.  Ansible, Chef, Puppet: инструменты для автоматизации развертывания и управления конфигурациями систем.  4. Платформы для облачных сервисов  Amazon Web Services (AWS): широкий спектр облачных услуг, включая вычислительные ресурсы, хранилища и управление данными.  Microsoft Azure: облачная платформа от Microsoft, предлагающая возможности хостинга, баз данных и анализа данных.  Google Cloud Platform (GCP): облачные сервисы от Google, включая инструменты для машинного обучения, аналитики и хранения данных.  Заключение  Выбор платформы для создания, исполнения и управления информационной системой зависит от многих факторов, включая требования к функциональности, масштабируемости, доступности и производительности системы. Существует множество инструментов и технологий, которые могут использоваться для эффективной разработки и поддержки информационных систем, и каждая организация должна найти подходящие решения, соответствующие её уникальным потребностям. |  |
| 12.11 | Отслеживание прогресса: регулярная оценка выполнения задач и сравнении с планом.  Управление бюджетом: мониторинг расходов и финансовых ресурсов, чтобы избежать перерасхода.  Отчётность: предоставление отчётов о состоянии проекта заинтересованным сторонам.  5. Завершение проекта  Этап, на котором проект закрывается и подводятся итоги. Включает:  Оценка результатов: анализ достигнутых результатов по сравнению с первоначальными целями.  Документирование уроков: сбор информации о том, что работало хорошо, а что нет, чтобы использовать эти данные в будущих проектах.  Официальное закрытие: завершение всех работ, передача итогового продукта заказчику и закрытие финансовых обязательств.  Важные методологии управления проектами  Кроме указанных процессов, существует несколько популярных методологий для управления проектами разработки, которые могут использоваться в зависимости от особенностей проекта:  Agile: гибкий подход, акцентирующий внимание на итеративной разработке и постоянном взаимодействии с заказчиком.  Scrum: методология Agile, использующая короткие итерации и специальные роли (Scrum Master, Product Owner).  Waterfall: традиционный подход, где проект делится на чёткие этапы, которые следует завершить до перехода к следующему.  Эти процессы и методологии помогают обеспечить успешное управление проектами, оптимизировать ресурсы и достигать поставленных целей. |  |
| 13.11 | Управление проектом разработки — это сложный, многоуровневый процесс, который включает в себя различные методологии и техники для достижения поставленных целей в рамках ограничений по времени, бюджету и качеству. Рассмотрим основные процессы управления проектом, входящие в этот цикл.  1. Инициация проекта  На этом этапе происходит формирование идеи проекта и определение его основных характеристик. Включает:  Определение цели проекта: чёткое формулирование того, что должно быть достигнуто.  Анализ заинтересованных сторон (стейкхолдеров): выявление всех, кто будет затронут проектом, и понимание их потребностей и ожиданий.  Создание устава проекта: официальная документация, описывающая проект, его цели и участников.  2. Планирование  Ключевой этап, на котором разрабатывается детальный план выполнения проекта. Основные аспекты включают:  Определение объема проекта: формирование списка всех deliverables, которые необходимо создать.  Составление расписания: определение временных рамок для выполнения задач (Использование сетевых графиков, диаграмм Ганта и др.).  Оценка ресурсов: определение необходимых ресурсов (человеческих, материальных, финансовых).  Управление рисками: идентификация потенциальных рисков, оценка их воздействия и разработка стратегий их минимизации.  3. Исполнение  На этом этапе происходит фактическая реализация плана. Включает:  Координация работы команды: обеспечение коммуникации и сотрудничества участников проекта.  Контроль качества: применение методов для обеспечения соответствия deliverables стандартам качества.  Управление изменениями: обработка запросов на изменения и адаптация плана проекта при необходимости. |  |
| 14.11 | 4. Мониторинг и контроль Важный процесс, который проходит параллельно с исполнением проекта, включает:  Проектирование, разработка и тестирование информационных систем (ИС) — это множество взаимосвязанных процессов и этапов, для каждого из которых существуют специфические методы и средства. Давайте разберем их подробнее.  Методы проектирования информационных систем  Моделирование на основе объектов (Object-Oriented Modeling):  Использует абстракцию объектов для представления компонентов системы.  Подходы: UML (Unified Modeling Language), ER-диаграммы (Entity-Relationship).  Функциональное моделирование (Functional Modeling):  Ориентировано на описание функций, которые система должна выполнять.  Методологии: DFD (Data Flow Diagram), IDEF (Integration DEFinition) и другие.  Системный подход (System Approach):  Рассматривает систему как целостное соединение многих компонентов.  Используется для выявления взаимосвязей между элементами системы.  Сетевое моделирование (Network Modeling):  Анализирует взаимодействие между различными узлами системы.  Используется для проектирования распределенных систем.  Методы разработки программного обеспечения  Методология Agile:  Итерационный подход к разработке, который позволяет гибко управлять изменениями.  Содержит практики: Scrum, Kanban.  Методология Waterfall (Каскадная модель):  Линейный подход, где каждый этап разработки должен завершиться перед началом следующего.  Простой и понятный, но менее гибкий.  RAD (Rapid Application Development):  Ориентирован на быстрое создание приложений с использованием прототипирования.  Стремление к быстрому получению обратной связи от пользователей.  Extreme Programming (XP):  Фокусируется на техническом совершенствовании и выполнении требований клиентов через частые итерации и непрерывное тестирование.  Спиральная модель:  Сочетает проектирование и разработку с систематическим управлением рисками.  Особое внимание в этой методологии уделяется анализу рисков на всех этапах.  Средства проектирования  CASE-средства (Computer-Aided Software Engineering):  Программы, помогающие в автоматизации процессов проектирования и разработки, такие как Rational Rose, Visual Paradigm.  Системы управления проектами:  Инструменты для планирования и мониторинга: JIRA, Trello, Asana.  Gantt и Pert-диаграммы:  Для визуализации графиков выполнения задач и распределения ресурсов.  Требования и спецификации:  Использование инструментов для сбора и управления требованиями, таких как Confluence и Reqview.  Методы тестирования информационных систем  Тестирование на основе требований (Requirements-Based Testing):  Подход, при котором тесты разрабатываются на основе чётко определённых требований к системе.  Тестирование на основе риск-менеджмента (Risk-Based Testing):  Приоритет задаётся тестированию тех компонентов системы, которые имеют наибольший риск для проекта.  Автоматизированное тестирование:  Использование специализированных инструментов (например, Selenium, JUnit) для автоматизации проверок.  Интеграционное тестирование:  Процесс проверки взаимодействия между модулями системы.  Регрессионное тестирование:  Обеспечение того, что изменения в коде не повлияли на существующую функциональность.  Тестирование производительности:  Оценка систем под нагрузкой для определения их производительности и устойчивости (например, с помощью JMeter или LoadRunner).  Заключение  Комбинирование этих методов и средств в пределах жизненного цикла разработки ИС позволяет эффективно управлять проектами, снижать риски и повышать общее качество конечного продукта. Успех разработки во многом определяется правильным выбором методологий, оптимальным использованием средств и тщательным проведением тестирования. |  |
| 15.11 | Национальная и международная система стандартизации и сертификации  1. Национальная система стандартизации:  Национальная система стандартизации управляется государственными органами и направлена на разработку и внедрение стандартов, которые обеспечивают качество, безопасность и совместимость продукции на внутреннем рынке. Основные элементы:  Стандарты: Документы, содержащие требования, рекомендации или характеристики, которые применяются к процессам, продуктам или системам. В России ключевым организацией является Росстандарт.  Сертификация: Процесс подтверждения соответствия продукции установленным стандартам. Включает в себя проверки и испытания для подтверждения качества.  Нормативные документы: Включают ГОСТ (государственные стандарты), технические условия и другие регуляторные акты, обязательные для применения.  2. Международная система стандартизации:  На глобальном уровне стандартизация осуществляется через международные организации, такие как ISO (International Organization for Standardization) и IEC (International Electrotechnical Commission). Основные моменты:  Международные стандарты: Эти документы разрабатываются с учетом интересов множества стран. Примером может служить ISO 9001, которая фокусируется на управлении качеством.  Унификация требований: Обеспечивает совместимость и взаимозаменяемость продукции на международном уровне, что упрощает торговые отношения между странами.  Сертификация за пределами государственной юрисдикции: Сертификация продукции часто требует соблюдения международных стандартов, что также делает продукцию более конкурентоспособной на глобальном рынке.  Система обеспечения качества продукции  Система обеспечения качества — это комплекс мероприятий по управлению и контролю качества на всех этапах жизненного цикла, от проектирования до производства и обслуживания. Основные аспекты:  Управление качеством: Включает разработку стандартов и методик, обучение персонала, планирование и контроль.  Документация: Ведется учёт всех процедур и результатов проверок, что обеспечивает прозрачность и возможность отслеживания качества.  Внедрение системы менеджмента качества (СМК): Например, внедрение ISO 9000, которая основывается на принципах управления качеством.  Методы контроля качества  Контроль качества — это процесс, направленный на обеспечение соответствия продукции установленным стандартам и требованиям. Основные методы контроля:  Входной контроль:  Проводится на входе материалов и комплектующих. Оценивается соответствие входящих ресурсов стандартам.  Текущий контроль:  Оценка качества в процессе производства. Включает выборочный контроль, а также использование статистических методов для анализа производственных процессов (например, контрольные карты).  Выходной контроль (приёмка):  Финальная проверка готовой продукции перед поставкой. Продукция должна соответствовать всем стандартам и требованиям спецификаций.  Системы управления качеством:  Внедрение таких систем, как TQM (Total Quality Management), которые ставят целью постоянное улучшение качества на всех уровнях организации.  Статистический контроль процессов (SPC):  Использует статистические методы для мониторинга и контроля производственных процессов с целью выявления и устранения отклонений.  Аудит качества:  Позволяет оценить соответствие установленным стандартам. Может быть как внутренним, так и внешним.  Испытания и оценка продукции:  Проведение лабораторных испытаний на соответствие спецификациям, нормам и стандартам.  Заключение  Национальная и международная системы стандартизации и сертификации играют критическую роль в обеспечении качества продукции, повышении ее конкурентоспособности и защите интересов потребителей. Методы контроля качества помогают выявлять и предотвращать недостатки на всех этапах жизненного цикла продукта, что, в конечном итоге, способствует повышению общей эффективности производства и удовлетворенности клиентов. |  |
| 16.11  17.11 | ВЫХОДНОЙ ДЕНЬ  ВЫХОДНОЙ ДЕНЬ |  |
| 18.11 | Сервисно-ориентированные архитектуры (SOA - Service-Oriented Architecture) представляют собой подход к проектированию и построению программных систем, который подразумевает использование сервисов как основных строительных блоков. Эта архитектура фокусируется на разработке приложений и систем, состоящих из независимых, взаимосвязанных сервисов, которые могут взаимодействовать друг с другом через стандартные интерфейсы и протоколы.  Основные характеристики сервисно-ориентированной архитектуры  Универсальные сервисы:  Сервисы в SOA могут быть использованы повторно и интегрированы в различные приложения. Они представляют собой отдельные функциональные единицы, которые могут выполнять определенные задачи.  Стандартизированные интерфейсы:  Взаимодействие между сервисами осуществляется через четко определенные интерфейсы, обычно основанные на веб-службах (например, SOAP, REST), что позволяет разным системам общаться между собой независимо от языка программирования и платформы.  Независимость и гибкость:  Каждый сервис может быть разработан, развернут и изменён независимо. Это способствует улучшению гибкости, так как изменения в одном сервисе не требуют изменений в других.  Масштабируемость:  SOA позволяет легко масштабировать отдельные сервисы в зависимости от нагрузки. Сервисы могут быть повторно использованы, что способствует более эффективному распределению ресурсов.  Интеграция:  Сервисно-ориентированные архитектуры обеспечивают легкость интеграции различных систем, как внутренних, так и внешних, предоставляя возможности для использования существующих решений.  Преимущества архитектуры SOA  Повторное использование компонентов:  Сервисы могут использоваться в различных приложениях, что снижает затраты на разработку и время вывода новых продуктов на рынок.  Гибкость и адаптируемость:  SOA позволяет легко изменять и адаптировать сервисы в ответ на изменяющиеся бизнес-требования, что делает архитектуру более устойчивой к изменениям.  Повышение производительности:  За счёт распараллеливания задач и возможности распределения нагрузки на несколько сервисов, производительность системы в целом возрастает.  Улучшение качества обслуживания:  SOA обеспечивает более высокое качество взаимодействия между системами, так как все сервисы взаимодействуют по строго установленным протоколам.  Недостатки и вызовы  Сложность управления:  Администрирование множества независимых сервисов может быть сложным процессом, требующим дополнительных ресурсов и инструментов для мониторинга и управления.  Производительность:  Взаимодействие между сервисами через сети может значительно снизить производительность из-за задержек, особенно в высоконагруженных системах.  Безопасность:  Уязвимости могут возникать при взаимодействии между различными сервисами, и это требует дополнительных мер безопасности.  Требования к разработке:  Разработка сервисов требует новых навыков и знаний от команды, что может привести к увеличению времени и затрат на обучение.  Применение SOA  Сервисно-ориентированные архитектуры активно применяются в сфере бизнеса и технологий. Вот несколько примеров использования SOA:  Финансовые службы: Банк может использовать сервисы для управления транзакциями, клиентскими данными и отчетностью, обеспечивая при этом интеграцию с внешними системами.  Электронная коммерция: Онлайн-магазины могут объединять разные сервисы (оплата, доставка, управление складом) для создания целостного решения, которое легко настраивается под бизнес-потребности.  Медицинские системы: В здравоохранении SOA может использоваться для интеграции различных информационных систем, обеспечивая обмен данными между больницами, лабораториями и страховыми компаниями.  Заключение  Сервисно-ориентированная архитектура предоставляет мощные инструменты для построения гибких, масштабируемых и легко интегрируемых систем. Однако ее успешная реализация требует четкого понимания бизнес-процессов, технической инфраструктуры и архитектурных паттернов, чтобы минимизировать риски и сложности, возникающие при использовании таких подходов. |  |
| 19.11 | Рассмотрение всех возможных вариантов и получение наилучшего решения на основе анализа и интересов клиента является ключевым аспектом эффективного принятия решений в различных сферах, будь то бизнес, право, личные отношения или другие области.  1. Понимание интересов клиента  Прежде всего, важно четко определить и понять интересы и потребности клиента. Это может включать в себя беседы, анкетирование или использование различных методов сбора информации. Зная, что именно ищет клиент, мы можем более эффективно рассматривать варианты решений.  2. Анализ всех возможных вариантов  Важно не ограничиваться только очевидными решениями. Ведение глубокого анализа включает в себя сравнение различных альтернатив, оценку их плюсов и минусов, а также прогнозирование возможных последствий каждого варианта. Это дает возможность избежать спешки и принимания решений на основании недостаточной информации.  3. Сравнение и выбор наилучшего решения  После выявления и анализа всех вариантов необходимо провести их сравнение, основываясь на заранее определенных критериях — таких как стоимость, срок реализации, эффективность и риски. Это позволяет объективно оценить каждое решение и выбрать то, которое наилучшим образом отвечает интересам клиента.  4. Учет долгосрочных последствий  Краткосрочные выгоды могут показаться привлекательными, однако важно также учитывать долгосрочные последствия. Здесь стоит использовать SWOT-анализ (сильные и слабые стороны, возможности и угрозы) для выявления потенциальных сценариев.  5. Обратная связь и корректировка  После принятия решения и его реализации, необходимо запрашивать обратную связь от клиента. Это поможет понять, насколько решение было успешным и соответствует ли оно ожиданиям. В случае необходимости можно провести корректировку или внести изменения.  6. Создание доверительных отношений  Рассматривая все возможные варианты и действуя в интересах клиента, вы не только выстраиваете доверительные отношения, но и повышаете свою репутацию как профессионала. Клиенты ценят тот подход, при котором их мнение учитывается и они вовлечены в процесс принятия решений.  7. Примеры успешных практик  В бизнесе компании, которые активно вовлекают клиентов в процесс принятия решений и учитывают их мнения, как правило, достигают лучших результатов. Например, фирмы, использующие краудсорсинг для сбора идей от клиентов, могут разработать более успешные продукты, чем те, кто полагается только на внутренние идеи.  Заключение  Таким образом, важность рассмотрения всех возможных вариантов и получения наилучшего решения на основе анализа и интересов клиента нельзя переоценить. Это не только способствует улучшению качества принимаемых решений, но и создает основу для долгосрочных, взаимовыгодных отношений. Обеспечение внимания к интересам клиента и учет всех возможных альтернатив формирует конкурентное преимущество и ведет к устойчивому успеху как для бизнеса, так и для индивидуальных практиков. |  |
| 20.11 | Системный анализ — это многогранный и комплексный процесс, целью которого является изучение и оптимизация сложных систем. Он охватывает теории, методы и инструменты, которые используются для анализа, проектирования и управления системами. Рассмотрим основные понятия системного анализа более подробно.  1. Система  В системном анализе система определяется как совокупность взаимосвязанных элементов, которые взаимо действуют для достижения общей цели. Системы могут быть разного вида:  Физические (например, машины, заводы);  Социальные (например, организации, общества);  Экономические (например, рынки, финансовые системы);  Экологические (например, экосистемы).  2. Элементы системы  Элементы — это отдельные компоненты системы, которые взаимодействуют между собой. Каждая система состоит из:  Подсистем — меньших, полностью функциональных систем внутри более крупной системы.  Входов и выходов — входы представляют собой ресурсы, используемые системой, а выходы — результаты, которые производит система.  3. Структура системы  Структура системы относится к организации ее элементов и взаимосвязей между ними. Она включает в себя:  Иерархию элементов — показывающую, как элементы расположены друг по отношению к другу.  Связи — определяющие, как элементы взаимодействуют друг с другом.  4. Процессы  Процессы — это действия или операции, которые выполняются внутри системы, чтобы преобразовать входные данные в выходные. Каждый процесс может быть детализирован на более мелкие шаги, и их эффективность критична для успешного функционирования всей системы.  5. Цели и задачи  Каждая система создается с определенной целью. Как правило, цели могут быть краткосрочными и долгосрочными. Задачи — это конкретные действия или шаги, которые необходимо выполнить для достижения этих целей.  6. Окружение системы  Окружение включает в себя все внешние факторы, которые могут оказывать влияние на систему. Это могут быть:  Экономические факторы,  Социальные тренды,  Технологические изменения и другие аспекты, которые могут как поддерживать, так и ограничивать функционирование системы.  7. Модель системы  Модель — это упрощенное представление системы, которое помогает изучить и проанализировать ее поведение. Модели могут быть:  Физическими (например, макеты),  Математическими (например, уравнения),  Графическими (например, диаграммы).  8. Методы и инструменты  Системный анализ предполагает использование различных методов и инструментов для решения задач. К ним относятся:  SWOT-анализ (анализ сильных и слабых сторон, возможностей и угроз),  Моделирование (сценарное или математическое моделирование),  Анализ требований и другие методы, помогающие в анализе системы.  9. Обратная связь  Обратная связь — это информация о работе системы, которая позволяет оценить ее эффективность и вносить необходимые коррективы. Это важный компонент для управления системой и ее адаптации к изменяющимся условиям.  10. Оптимизация  Оптимизация включает в себя процесс улучшения функционирования системы с целью достижения наилучших результатов. Это может касаться улучшения качества, снижения затрат, увеличения эффективности и т.д.  Заключение  Системный анализ — это ценный инструмент для решения комплексных проблем в различных областях, включая управление проектами, разработку программного обеспечения и оценку бизнес-процессов. Понимание основных понятий системного анализа позволяет лучше подходить к проектированию и управлению любой системой, обеспечивая эффективное достижение поставленных целей. |  |
| 21.11 | Реинжиниринг бизнес-процессов (РБП) представляет собой стратегический подход, нацеленный на радикальное переосмысление и кардинальное улучшение существующих бизнес-процессов с целью достижения значительных улучшений в критически важных показателях, таких как стоимость, качество, обслуживание и скорость. Процесс реинжиниринга, как правило, включает в себя пересмотр и переработку структуры, методов и системы работы компании.  Основные компоненты реинжиниринга бизнес-процессов  Представление о процессе  Прежде всего, необходимо четкое понимание текущих бизнес-процессов. Это включает в себя анализ существующих потоков работ, ресурсов и технологий, которые используются в организации.  Цели реинжиниринга  Определение четких и измеримых целей - важная часть процесса. Это могут быть, например, улучшение обслуживания клиентов, сокращение времени на выполнение заказов или снижение операционных затрат.  Изменение структуры  Часто реинжиниринг включает в себя изменение организационной структуры, например, путем создания кросс-функциональных команд, которые могут более эффективно справляться с задачами, пересекающими различные области.  Инновационные технологии  Использование новых технологий и инструментов может значительно повысить эффективность бизнес-процессов. Это включает в себя автоматизацию, внедрение программного обеспечения для управления проектами и использования аналитики больших данных.  Фокус на клиенте  Реинжиниринг бизнес-процессов должен быть ориентирован на потребности клиентов. Это предполагает, что процессы и услуги должны создаваться с акцентом на улучшение клиентского опыта.  Этапы реинжиниринга бизнес-процессов  Идентификация процессов для реинжиниринга  Определение тех процессов, которые требуют улучшения. Это может включать в себя опросы сотрудников, анализ данных производительности и изучение отзывов клиентов.  Анализ текущего состояния  Использование различных методов, таких как картирование процессов, для анализа существующей ситуации. Это позволяет выявить узкие места, дублирование усилий и другие недостатки.  Проектирование новых процессов  Разработка новых, более эффективных процессов, которые устраняют ранее выявленные проблемы. Это может включать изменение последовательности действий, внедрение новых роли или использование новых технологий.  Внедрение изменений  Переход к новой системе требует тщательной подготовки, включая обучение сотрудников, изменение стандартов операционной деятельности и коммуникацию с заинтересованными сторонами.  Оценка результатов  После внедрения новых процессов необходимо провести оценку их эффективности. Это подразумевает использование заранее определенных метрик и показателей, чтобы определить, достигнуты ли цели реинжиниринга.  Примеры успешного реинжиниринга  Ford Motor Company: В 1980-х годах компания реализовала реинжиниринг, сосредоточившись на переработке производственных процессов, что позволило оптимизировать запасы и сократить время сборки автомобилей.  CitiGroup: В отрасли финансовых услуг CitiGroup внедрила реинжиниринг для упрощения своих операций. Это позволяло им значительно улучшить качество предоставляемых услуг и скорость обслуживания клиентов.  Преимущества и риски  Преимущества:  Улучшение эффективности и производительности;  Снижение затрат;  Повышение уровня обслуживания клиентов;  Увеличение гибкости и способности к адаптации.  Риски:  Сопротивление изменениям со стороны сотрудников;  Потеря знаний и опыта в процессе изменения;  Возможные временные потери в производительности в переходный период.  Заключение  Реинжиниринг бизнес-процессов может стать мощным инструментом для организаций, стремящихся улучшить свою эффективность и конкурентоспособность. Однако, как и любые серьезные изменения, этот процесс требует тщательного планирования, анализа и управления. Успех реинжиниринга во многом зависит от участия и поддержки со стороны всех уровней организации, а также от способности адаптироваться к новым условиям. |  |
| 22.11 | Система обеспечения качества продукции (СОКП) представляет собой интегрированную структуру, направленную на управление качеством на всех этапах жизненного цикла продукции — от разработки и производства до реализации и обслуживания. Главная цель СОКП — гарантировать, что продукция или услуги соответствуют установленным стандартам качества и удовлетворяют требованиям потребителей.  Основные элементы системы обеспечения качества  Политика качества  Определяет стратегические цели в области качества и стремления организации к обеспечению высоких стандартов. Политика качества должна быть понятна всем сотрудникам и служить основой для разработки других элементов СОКП.  Планирование качества  В этом этапе разрабатываются документы, описывающие процессы, необходимые для достижения целей качества. Это может включать в себя схемы обеспечения качества, контрольные планы, графики испытаний и процедуры.  Управление ресурсами  Успешное выполнение планов по обеспечению качества требует наличия необходимых ресурсов. Это включает как человеческие ресурсы (подбор и обучение персонала), так и технические (оборудование, технологии).  Контроль процесса  На этом этапе осуществляется мониторинг и измерение производственных процессов для обеспечения их соответствия стандартам качества. Это может включать регулярные проверки, инспекции и тестирования в процессе производства.  Управление документами и записями  Важно иметь четкую систему документирования всех процессов и процедур, связанного с качеством. Это не только упрощает контроль и анализ, но и способствует соблюдению стандартов и норм.  Анализ данных  Сбор и анализ данных о качестве позволяют выявлять тенденции, проблемы и возможности для улучшения. Это может включать в себя статистический анализ, аудиты качества и обратную связь от клиентов.  Устранение несоответствий  В случае выявления несоответствий, необходимо разработать и внедрить коррекционные действия. Это включает в себя анализ причин несоответствия и реализацию мер для их устранения.  Улучшение системы  Постоянное улучшение — ключевая составляющая любой эффективной системы обеспечения качества. Это может осуществляться через регулярные переоценки процессов, внедрение новых технологий и методов, а также путем адаптации к изменениям на рынке. |  |
| 23.11  24.11 | ВЫХОДНОЙ ДЕНЬ  ВЫХОДНОЙ ДЕНЬ |  |
| 25.11 | Стандарты систем обеспечения качества  Наиболее ampliamente используемым стандартом является ISO 9001, который устанавливает требования к системам управления качеством. Он охватывает все аспекты организации и требует участия руководства в вопросах качества, а также ориентации на клиента.  Преимущества системы обеспечения качества  Повышение удовлетворенности клиентов  Нормы и процессы, направленные на качество, обеспечивают соответствие ожиданиям клиентов, что, в свою очередь, способствует повышению их удовлетворенности и лояльности.  Снижение затрат  Благодаря оптимизации процессов и устранению несоответствий, системы обеспечения качества могут значительно снизить затраты на исправление ошибок и возврат продукции.  Улучшение внутренней эффективности  Стандартизированные процессы способствуют более эффективному взаимодействию между отделами и повышению общей производительности.  Конкурентные преимущества  Компании с развитыми системами управления качеством могут выделяться на рынке, предлагая более надежные и качественные продукты или услуги, что привлекает больше клиентов.  Культура качества  Внедрение системы обеспечения качества способствует формированию в организации культуры, ориентированной на постоянные улучшения и удовлетворение потребностей клиентов.  Заключение  Система обеспечения качества продукции является важнейшим аспектом успешной деятельности любой организации. Она не только позволяет обеспечить высокие стандарты качества, но и способствует созданию культуры постоянного улучшения, что жизненно важно для бизнеса в условиях конкурентного рынка. Эффективное управление качеством требует постоянного внимания, анализа и адаптации к меняющимся условиям для достижения максимальной удовлетворенности клиентов и устойчивости на рынке. |  |
| 26.11 | Контроль качества — это комплекс мероприятий, направленных на обеспечение соответствия продукции или услуг установленным требованиям и стандартам. В зависимости от уровня сложности и специфики производства, применяются различные методы контроля качества. Рассмотрим наиболее распространенные из них, в том числе их применение в соответствии с базовыми стандартами, как ISO 9001 и другими.  1. Визуальный контроль  Описание:  Визуальный контроль включает в себя осмотр продукции на предмет видимых дефектов, соответствия технологическим требованиям и стандартам.  Применение:  Для проверки внешнего вида продукции.  Для выявления очевидных дефектов или повреждений после производства.  Используется на всех этапах — от поступления материалов до финальной проверки готовой продукции.  2. Измерительный контроль  Описание:  Предполагает использование измерительных инструментов для проверки характеристик продукции на соответствие стандартам.  Применение:  Проверка геометрических размеров, весов, объемов и других физических параметров.  Использование калибров, штангенциркулей, микрометров и других средств измерений.  3. Статистический контроль процессов (SPC)  Описание:  Метод основан на статистическом анализе данных о процессе производства и его выходах в реальном времени с целью выявления и устранения причин вариаций.  Применение:  Используется для мониторинга процессов и определения стабильности и управляемости.  Установление контрольных границ и анализ отклонений.  4. Приемочный контроль  Описание:  Система проверки качества, которая применяется к партиям продукции или услуг. Проверка может осуществляться на основе статистических методов и выбрать случайные образцы для тестирования.  Применение:  Широко используется в условиях массового производства.  Определение приемлемого уровня качества (AQL) для партий товаров и принятие решений о приемке или отклонении.  5. Испытания и испытания на разрушение  Описание:  Методы, используемые для оценки прочности, выносливости и других характеристик продукции под действием различных условий.  Применение:  Проверка материалов и компонентов на соответствие специфическим стандартам (например, термостойкость, устойчивость к коррозии).  Используется для оценки надежности и долговечности продукции.  6. Аудиты качества  Описание:  Систематическая проверка процессов и систем управления качеством на соответствие стандартам и требованиям.  Применение:  Внутренние и внешние аудиты для оценки эффективности системы управления качеством.  Использование результатов аудитов для выявления возможностей для улучшения и обучения.  7. Анализ корневых причин (RCA)  Описание:  Метод, предполагающий выявление основных причин проблем с качеством и их устранение.  Применение:  Используется для решения проблем, возникающих в процессе производства.  Основывается на систематическом анализе проблем через различные подходы, такие как "5 почему".  8. Управление изменениями  Описание:  Метод, использующийся для контроля изменений в процессах или продукции, чтобы гарантировать, что эти изменения не влияют на уровень качества.  Применение:  Формализованный процесс, который включает документы и процедуры для корректного управления изменениями.  9. Обратная связь от клиентов  Описание:  Сбор и анализ отзывов и замечаний от клиентов.  Применение:  Важный элемент в системе обеспечения качества, позволяющий своевременно реагировать на потребности и претензии клиентов.  Используется для разработки последующих стратегий улучшения качества.  Заключение  Методы контроля качества играют ключевую роль в поддержании стандартов и повышении удовлетворенности клиентов. Выбор конкретного метода зависит от специфики производства и продукта, а также от стандартов, применяемых в соответствующей отрасли. Комплексное применение различных методов позволяет достичь высоких стандартов качества и обеспечить конкурентоспособность на рынке. |  |
| 27.11 | Объектно-ориентированное программирование (ООП) — это парадигма программирования, основанная на концепции "объектов", которые могут содержать как данные, так и код. Данные описываются с помощью атрибутов (или свойств), а код — с помощью методов (или функций). ООП предоставляет мощные механизмы для разработки сложных программных систем, позволяя создать более структурированный и поддерживаемый код. Рассмотрим ключевые концепции и основные принципы ООП.  Основные концепции ООП  Объекты:  Объект — это экземпляр класса, который объединяет состояние (данные, или атрибуты) и поведение (методы, или функции). Например, в программе, управляющей библиотекой, объектом может быть "Книга", содержащая атрибуты, такие как "название", "автор", "год издания", и методы, такие как "выдать" или "вернуть".  Классы:  Класс — это шаблон для создания объектов. Он определяет, какие атрибуты и методы будут у объектов этого класса. Классы позволяют инкапсулировать данные и поведение в одном модуле. Например, класс "Автомобиль" может иметь атрибуты, такие как "цвет", "марка", и методы, такие как "запускать" или "остановить".  Наследование:  Наследование — это механизм, позволяющий создавать новые классы на основе уже существующих. Новый класс (наследник) наследует свойства и методы родительского класса, что позволяет повторно использовать код и уменьшать дублирование. Например, класс "Спортивный автомобиль" может наследоваться от класса "Автомобиль" и добавлять уникальные атрибуты, такие как "максимальная скорость".  Полиморфизм:  Полиморфизм позволяет объектам разных классов обрабатывать данные по одному и тому же интерфейсу, что упрощает код и делает его более гибким. В ООП это обычно реализуется через переопределение методов. Например, метод "запуск" может быть реализован по-разному в классах "Бензиновый автомобиль" и "Электрический автомобиль".  Инкапсуляция:  Инкапсуляция — это сокрытие внутреннего состояния объекта и защита его от некорректного изменения извне. Это достигается за счет использования модификаторов доступа (например, private, protected, public). Инкапсуляция способствует созданию более безопасного и поддерживаемого кода, так как изменения в внутренней реализации не затрагивают интерфейс класса.  Принципы ООП  Согласованность: Объекты должны взаимодействовать друг с другом через строго определенные интерфейсы, что обеспечивает предсказуемость работы системы.  Модульность: Код разделяется на независимые модули (классы и объекты), что позволяет легче управлять его сложностью и тестировать его.  Повторное использование кода: возможность создания новых классов на основе существующих (наследование) и использования общих интерфейсов (полиморфизм) увеличивает скорость разработки и снижает количество ошибок.  Применение ООП  Объектно-ориентированное программирование широко используется в различных областях разработки программного обеспечения, включая:  Разработка приложений: Создание настольных и мобильных приложений, которым требуется хорошо структурированная архитектура.  Игровая разработка: Моделирование игровых объектов, их свойств и взаимодействия.  Веб-разработка: Выстраивание архитектуры серверной и клиентской части веб-приложений.  Системное программирование: Разработка сложных систем, таких как средства автоматизации и управления.  Примеры ООП  Рассмотрим небольшой пример на языке Python, который иллюстрирует основные концепции ООП.  # Определение класса "Животное"  class Animal:  def \_\_init\_\_(self, name):  self.name = name  def speak(self):  pass # Методы будут переопределены в дочерних классах  # Наследование класса "Кошка" от "Животное"  class Cat(Animal):  def speak(self):  return "Мяу"  # Наследование класса "Собака" от "Животное"  class Dog(Animal):  def speak(self):  return "Гав"  # Создание объектов  cat = Cat("Барсик")  dog = Dog("Рекс")  # Взаимодействие с объектами  print(cat.name + " говорит: " + cat.speak()) # Вывод: Барсик говорит: Мяу  print(dog.name + " говорит: " + dog.speak()) # Вывод: Рекс говорит: Гав  В этом примере мы создали базовый класс "Animal", который содержит метод speak, а затем реализовали два класса-наследника Cat и Dog, которые предоставляют свою реализацию этого метода.  Заключение  Объектно-ориентированное программирование является мощным инструментом для разработки программного обеспечения, предоставляя разработчикам способ организовывать код, облегчать его понимание и поддержку. ООП нацелено на создание более устойчивых, модульных и повторно используемых компонентов, что делает его одной из самых популярных парадигм программирования в мире.  Спецификации языка программирования, принципы создания графического пользовательского интерфейса (GUI).  Создание графического пользовательского интерфейса (GUI) является важной частью разработки программного обеспечения. Спецификации языка программирования и принципы создания пользовательского интерфейса играют ключевую роль в этом процессе. Рассмотрим основные аспекты, касающиеся спецификаций языков программирования и лучших практик в разработке GUI.  Спецификации языков программирования для разработки GUI  Разные языки программирования предоставляют различные возможности и инструменты для создания графических пользовательских интерфейсов. Рассмотрим несколько популярных языков и их спецификации:  Java:  Использует библиотеку Swing или JavaFX для создания GUI.  Swing предоставляет компоненты, такие как кнопки, текстовые поля и панели, которые могут быть скомбинированы для построения интерфейса.  JavaFX поддерживает более современные графические элементы и анимации, а также предоставляет возможность работы сFXML (разметка XML для описания интерфейса).  Python:  Поддерживает несколько библиотек для создания интерфейсов, таких как Tkinter, PyQt и Kivy.  Tkinter — стандартная библиотека для создания GUI, включенная в дистрибутив Python, и проста в использовании.  PyQt и Kivy предлагают более сложные возможности и подходят для создания кросс-платформенных приложений.  C#:  Использует Windows Forms или WPF (Windows Presentation Foundation) для разработки интерфейсов на платформе .NET.  Windows Forms подходит для создания простых приложений, тогда как WPF предлагает мощные средства для работы с графикой и анимацией.  JavaScript:  Веб-приложения используют JavaScript совместно с HTML и CSS для создания динамичных интерфейсов.  Библиотеки и фреймворки, такие как React, Angular и Vue.js, значительно упрощают разработку более сложных пользовательских интерфейсов.  Принципы создания графического пользовательского интерфейса (GUI)  Для создания эффективного и удобного интерфейса существуют несколько проверенных принципов:  1. Простота  Интерфейс должен быть интуитивно понятным. Чем проще его дизайн, тем легче пользователю выполнить необходимые действия. Избегайте избыточного количества элементов.  2. Консистентность  Все элементы интерфейса должны следовать единым стилям и поведению. Например, если кнопки работают определенным образом, они должны иметь одинаковый вид и поведение везде в приложении.  3. Обратная связь  Приложение должно предоставлять обратную связь пользователю. Например, кнопка должна изменять цвет при наведении, или показывать сообщение о сохранении данных.  4. Доступность  Обеспечьте доступность интерфейса для пользователей с ограниченными возможностями. Это включает использование контрастных цветов, увеличенный текст и поддержку экранных читалок.  5. Упрощение навигации  Хорошая структура навигации позволяет пользователю быстро находить нужные функции и информацию. Используйте меню, навигационные панели и хлебные крошки.  6. Группировка и иерархия  Связанные элементы интерфейса должны быть сгруппированы логически. Иерархия важна для представления структуры данных и функциональных возможностей.  7. Тестирование с пользователями  Проведение тестов с реальными пользователями позволяет выявить проблемы в интерфейсе и улучшить его на основе обратной связи.  Заключение  Разработка графического пользовательского интерфейса — это комбинация художественного и технического подходов. Знание спецификаций языков программирования и применение лучших практик интерфейсного дизайна позволит создать удобные и функциональные приложения, которые удовлетворяют потребности пользователей. Подход к разработке должен быть ориентирован на пользователя, чтобы обеспечить наилучший опыт работы с вашим приложением. |  |
| 28.11 | Спецификации языка программирования - это набор правил и синтаксических конструкций, определяющих структуру и поведение программного кода на определенном языке программирования. Спецификации также определяют возможности и ограничения языка программирования, типы данных, операторы, функции и другие элементы языка.  Принципы создания графического пользовательского интерфейса (GUI):  Простота использования: GUI должен быть интуитивно понятным для пользователя и обеспечивать простой и удобный способ взаимодействия с программой.  Согласованность: элементы интерфейса должны быть согласованными в своем дизайне, стилистике и поведении, чтобы создавать единое и последовательное впечатление.  Эффективность: GUI должен обеспечивать быстрый доступ к функциональности программы и удобные способы выполнения задач.  Гибкость: пользователь должен иметь возможность настраивать интерфейс под свои потребности и предпочтения.  Визуальная привлекательность: GUI должен быть эстетически приятным и привлекательным для пользователя.  Отзывчивость: интерфейс должен реагировать на действия пользователя немедленно, обеспечивая быструю обратную связь.  Понятность: элементы интерфейса должны быть четко и однозначно обозначены, чтобы пользователю было легко понять, что делает каждый элемент.  Масштабируемость: GUI должен быть масштабируемым, чтобы обеспечить удобство использования на различных устройствах и разрешениях экранов.  Доступность: GUI должен быть доступным для всех пользователей, включая людей с ограниченными возможностями, такими как слабовидящие или люди с ограниченными двигательными способностями.  Удобство навигации: пользователю должно быть легко найти нужные функции и перемещаться по интерфейсу без лишних усилий.  Консистентность: GUI должен сохранять одинаковый стиль и поведение на протяжении всего приложения, чтобы пользователь не запутался и легко ориентировался. |  |
| 29.11 | Поддержка мультиязычности: интерфейс должен поддерживать разные языки для обеспечения удобства использования приложения пользователями из разных стран и культур.  Понятность и информативность: интерфейс должен быть интуитивно понятным и содержать достаточное количество информации для пользователя, чтобы он мог легко освоить функционал приложения.  Возможность настройки: GUI должен предоставлять пользователю возможность настройки интерфейса в соответствии с его предпочтениями и потребностями.  Безопасность: интерфейс должен обеспечивать защиту данных и конфиденциальность пользователя, а также предоставлять необходимые средства для обеспечения безопасности при использовании приложения. |  |
| 30.11  01.12 | ВЫХОДНОЙ ДЕНЬ  ВЫХОДНОЙ ДЕНЬ |  |
| 02.12 | Визуальная ясность: элементы интерфейса должны быть понятными и ясно отображаться на экране.  Цвета, шрифты и изображения должны быть выбраны таким образом, чтобы облегчить восприятие информации.  Эффективность: интерфейс должен быть удобным для выполнения всех необходимых действий с минимальными усилиями со стороны пользователя. Ненужные или сложные элементы следует уменьшить или убрать.  Обратная связь: пользователю должно быть предоставлено информация о выполненных им действиях и текущем состоянии приложения. Это поможет пользователям понять, что происходит и что они могут ожидать.  Адаптивность: интерфейс должен быть адаптирован под разные разрешения экранов, устройства и потребности пользователей. Также важно учитывать доступность для пользователей с ограниченными возможностями.  Эти принципы помогают создать удобный, интуитивно понятный и эффективный графический пользовательский интерфейс для любого программного продукта. |  |
| 03.12 | Спецификации языков программирования включают синтаксис, стиль программирования, основные конструкции выбранного языка, описание переменных, организацию ввода-вывода, реализацию типовых алгоритмов.  Принципы создания графического пользовательского интерфейса (GUI) включают разработку в зависимости от используемых устройств ввода (тачскрин, мышь, клавиатура). Например, для сенсорного экрана (смартфоны и мобильные устройства высокого класса) требуются более крупные значки.  Для создания GUI в веб-приложениях используются технологии HTML (структура страниц), основы CSS (внешний вид) и основы JavaScript (функциональность элементов страницы).  Для работы с GUI в Python можно использовать библиотеку Tkinter, которая позволяет создавать приложения для Windows, mac OS и Linux.  Для этого нужно подключить библиотеку, создать главное окно приложения, в котором будут размещаться все графические элементы, добавить виджеты (визуальные элементы, выполняющие определённые действия) и создать главный цикл событий, который включает в себя все события, происходящие при взаимодействии пользователя с интерфейсом.  Спецификации языка программирования определяют правила написания кода на определенном языке, его синтаксис и семантику. Это включает в себя типы данных, операторы, функции, классы и другие элементы, которые могут использоваться при разработке программного обеспечения.  Принципы создания графического пользовательского интерфейса (GUI) включают в себя следующие основные принципы:  Простота использования: интерфейс должен быть интуитивно понятным и легким для использования даже без предварительного обучения. Пользователь должен легко понимать, как выполнять необходимые действия.  Единый стиль: все элементы интерфейса должны выглядеть согласованно и иметь одинаковое оформление. Это поможет создать цельное и удобное в использовании приложение. |  |
| 04.12 | Создание файлового ввода-вывода — это процесс организации ввода-вывода не на стандартные устройства, а в файл.  В языке C++ ввод-вывод осуществляется через объекты классов ifstream (для чтения данных) и ofstream (для вывода данных).  Эти классы объявлены в заголовочном файле fstream.  При работе с файлом можно выделить следующие этапы:  1. Создать объект класса fstream (возможно, ofstream или ifstream).  2. Связать объект класса fstream с файлом, который будет использоваться для операций ввода-вывода.  3. Осуществить операции ввода-вывода в файл.  4. Закрыть файл.  5. После окончания работы с файлами объекты нужно «закрыть», вызвав для них метод close().  Сегодня я провел целый день, работая над созданием файлового ввода-вывода для нашего проекта. Этот этап работы требовал от меня не только тщательной проработки кода, но и глубокого понимания принципов работы файловой системы.  В начале дня я провел необходимые исследования, изучив основные принципы работы файлового ввода-вывода и способы его реализации в нашем проекте. После этого приступил к написанию кода, который позволит работать с файлами, читать информацию из них и записывать в них данные.  На протяжении всего дня я тестировал код, чтобы убедиться, что он работает корректно и эффективно. Ошибки были обнаружены и исправлены, улучшения были внесены для оптимизации процесса работы с файлами.  В конце дня я создал подробную документацию по созданному файловому вводу-выводу, чтобы наши коллеги могли легко понять его работу и использовать его в своей работе.  В целом, этот день был очень продуктивным и насыщенным. Я узнал много нового о работе с файлами и повысил свои навыки программирования. Создание файлового ввода-вывода оказалось интересным и полезным опытом, который поможет улучшить наш проект и сделать его более эффективным. |  |
| 05.12 | Создание сетевого сервера и сетевого клиента — это процесс разработки сетевых приложений, которые обмениваются данными через сеть.  Сервер в такой схеме — это часть программной системы, модуль, который постоянно выполняется и ждёт запросов от клиентов. Когда запрос поступает, сервер его обрабатывает и выдаёт ответ.  Клиент — это модуль программы, который непосредственно взаимодействует с пользователем программы и в зависимости от его действий может инициировать соединение с сервером, чтобы произвести определённые операции.  Таким образом, сервер предоставляет ресурсы устройствам, подключённым через сеть, а клиент полагается на сервер для получения ресурсов.  Сервер:  import socket  server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  server\_socket.bind(('localhost', 5555))  server\_socket.listen(5)  print("Сервер запущен")  while True:      client\_socket, addr = server\_socket.accept()      print(f"Подключение от {addr}")      client\_socket.send("Привет, клиент!".encode())      client\_socket.close() |  |
| 06.12 | Клиент:  import socket  client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) client\_socket.connect(('localhost', 5555))  data = client\_socket.recv(1024) print(data.decode())  client\_socket.close()  Запустите сервер и клиент в разных окнах терминала. Когда клиент подключится к серверу, он получит сообщение "Привет, клиент!".  Это простой пример, но вы можете расширить его, добавив обработку различных команд, передачу файлов и другие функции.  Не забудьте обработать исключения и закрыть сокеты после использования, чтобы избежать утечек памяти. |  |
| 07.12  08.12 | ВЫХОДНОЙ ДЕНЬ  ВЫХОДНОЙ ДЕНЬ |  |
| 09.12 | Области применения языков программирования:  1. Веб-разработка  2. Мобильная разработка  3. Разработка игр  4. Разработка настольных приложений.  Наиболее популярными направлениями программирования являются в наше время следующие направления: веб-разработка; разработка мобильных приложений; разработка компьютерных игр; разработка приложений для десктопов.  Нельзя сказать, что за каждым из них жестко закреплены те или иные языки программирования.  Основные особенности и области применения включают в себя:  Широкий спектр применения: искусственный интеллект может использоваться во многих отраслях, таких как здравоохранение, финансы, транспорт, производство, образование и др.  Автоматизация процессов: благодаря искусственному интеллекту можно автоматизировать многие повседневные задачи, что помогает увеличить производительность и эффективность работы.  Принятие решений: искусственный интеллект может анализировать большие объемы данных и помогать принимать более обоснованные решения на основе этих данных.  Разработка продуктов и услуг: многие компании используют искусственный интеллект для разработки новых продуктов и услуг, которые более точно соответствуют потребностям пользователей.  Обработка естественного языка: искусственный интеллект также используется для обработки и анализа естественного языка, что помогает улучшить коммуникацию между людьми и компьютерами.  Развитие робототехники: искусственный интеллект играет ключевую роль в развитии робототехники, что позволяет создавать автономных роботов и механизмы, способные выполнять сложные задачи.  Персонализация: благодаря искусственному интеллекту компании могут создавать персонализированные рекомендации и предложения для своих клиентов, улучшая качество обслуживания и удовлетворение потребностей.  Таким образом, искусственный интеллект имеет широкий спектр применения и может быть использован в различных отраслях для улучшения работы и повышения эффективности бизнес-процессов. |  |
| 10.12 | Особенности программных средств, используемых в разработке информационных систем (ИС), могут включать:  Использование прикладных программ в качестве инструмента для обработки информации. Представление найденной или хранимой информации. Решение более общих задач, чем специализированное прикладное программное обеспечение. Наличие среды хранения и доступа к данным. Она должна обеспечивать уровень надёжности хранения и эффективность доступа, соответствующие области применения ИС. Простой и удобный интерфейс. Он должен предоставить конечному пользователю все необходимые для его работы функции, но не дать ему возможность выполнять какие-либо лишние действия. Использование различных программных приложений. Например, систем для генерации отчётов, аналитических систем, генерации персональных запросов, анализа и реализации отчётной документации.  Интеграция с другими системами: программные средства, используемые в разработке информационных систем, должны быть способны взаимодействовать с другими системами и программами, чтобы обеспечить эффективную работу всей системы.  Модульность: программные средства должны быть разделены на модули для удобства разработки, тестирования и сопровождения системы. Каждый модуль должен выполнять определенную функцию, быть независимым и легко заменяемым.  Безопасность: программные средства должны обеспечивать защиту информации от несанкционированного доступа, изменения или уничтожения. Для этого могут использоваться различные методы шифрования, аутентификации и контроля доступа. |  |
| 11.12 | Масштабируемость: программные средства должны быть способны масштабироваться в зависимости от объема данных и нагрузки на систему. Это позволит системе эффективно работать как в маломасштабном режиме, так и в крупных корпоративных сетях.  Мобильность: с учетом современных тенденций в развитии информационных технологий, программные средства должны быть адаптированы под мобильные устройства, чтобы пользователи могли получать доступ к информационной системе в любое время и в любом месте.  Гибкость: программные средства должны быть гибкими и легко настраиваемыми для решения различных задач и потребностей пользователей. Возможность расширения функционала и добавления новых модулей должна быть реализована без серьезных изменений в основной системе.  Производительность: разработанные программные средства должны обеспечивать высокую производительность при выполнении различных операций и обработке данных. Это достигается оптимизацией кода, использованием эффективных алгоритмов и архитектурных решений.  Поддержка: важным аспектом программных средств является их поддержка и обновление. Разработчики должны обеспечивать регулярные обновления и исправление ошибок для обеспечения безопасности и стабильной работы информационной системы. |  |
| 12.12 | Для создания графического пользовательского интерфейса (GUI) можно использовать различные языки программирования и библиотеки. Например:  Java. Объектно-ориентированный язык на основе классов. Java ориентирован на кросс-платформенный подход, имеет графическую библиотеку, которая поддерживает двухмерную и трёхмерную графику.  Python. Для работы с GUI в Python можно использовать библиотеки Tkinter, Kivy, Python QT или wxPython. Например, Tkinter позволяет создавать приложения для Windows, Mac OS и Linux.  С++. Для разработки ПО на языке С++ распространённым средством является кроссплатформенный инструментарий Qt.  Принципы создания GUI включают определение целей и исследование целевой аудитории, разработку визуального языка (цвета, шрифты) и создание пробных версий пользовательского интерфейса. Также при разработке GUI важно учитывать используемые устройства ввода (тачскрин, мышь, клавиатура).  Например, для сенсорного экрана (смартфоны и мобильные устройства высокого класса) требуются более крупные значки, а на десктопе можно взаимодействовать даже с мелкими иконками и другими элементами GUI при помощи мыши. |  |
| 13.12 | Программирование — это создание инструкций для компьютера, то есть программ.  Обычно оно выглядит как написание программного кода на каком-нибудь языке программирования.  С его помощью разработчики создают сайты, приложения и разнообразные цифровые сервисы.  В более широком смысле программирование — любое создание инструкций для исполнителя.  Например, мама показывает ребенку, как сделать себе бутерброд. С компьютерным программированием так же, только инструкции пишут в виде кода на специальных языках. А исполнитель — какое-либо устройство.  Когда человек программирует, он, по сути, переводит задачи для машины на язык, который ей понятен. Инструкции, написанные на этом языке, могут запускаться и выполняться — компьютер им следует. Так получаются программы.  Для чего нужно программирование:  С помощью программирования созданы все сайты, сервисы, операционные системы и приложения, которыми мы пользуемся. Правда, еще есть подход zero-code, но о нем мы расскажем позже. Сейчас важно, что без программирования не получилось бы ни Windows, ни Google |  |
| 14.12 | Отчет по практике должен содержать следующие разделы:  Титульный лист - содержит название учебного заведения, отделения, направления подготовки, тему практики, ФИО студента, научного руководителя, дату.  Содержание - перечень всех разделов и подразделов отчета с указанием номеров страниц.  Введение - краткое описание цели и задачи практики, контекст проведения практики, актуальность темы.  Основная часть - самый обширный раздел отчета, включающий в себя описание проведенной работы, методы исследования, результаты и их анализ.  Заключение - суммирование основных результатов практики, выводы и рекомендации.  Список использованной литературы - перечень всех источников, использованных в работе.  Приложения (если есть) - дополнительные материалы, которые могут быть полезны для более глубокого понимания темы практики.  Касательно материала, необходимо включить достаточно информации для полного понимания темы практики, но при этом не утопить отчет в излишнем количестве деталей.  Рекомендуется подробно изложить основные пункты и результаты работы, описать методику исследования, но не углубляться в каждую мелочь.  В то же время, отчет должен быть структурированным, логичным и четким, чтобы читателю было легко ориентироваться в информации. Каждая глава и раздел должны быть ясно обозначены и разделены друг от друга.  Подводя итог, отчет по практике должен быть комплексным документом, отражающим всю проведенную работу студента и результаты его усилий.  Он должен содержать достаточно информации для понимания проведенной работы, быть структурированным и легким для восприятия читателем.  Важно следовать указанным выше разделам и рекомендациям для создания качественного отчета по практике. |  |

## Подпись руководителя и печать профильной

**организации**

**(ставится в последний день практики)**

## ВЫВОДЫ И ПРЕДЛОЖЕНИЯ

Примечание: в этом разделе студент делает выводы о работе организации, оценивает эффективность работы по основным сферам деятельности, выделяет новое, прогрессивное, вносит предложения по улучшению работы предприятия.

## ЗАКЛЮЧЕНИЕ РУКОВОДИТЕЛЯ ПРОФИЛЬНОЙ ОРГАНИЗАЦИИ ПО ПРАКТИЧЕСКОЙ ПОДГОТОВКЕ

В заключении необходимо указать:

а) отношение практиканта к работе (интерес, инициатива, самостоятельность, исполнительность, дисциплинированность и т.д.);

б) недостатки в теоретической подготовке студента, проявившиеся во время практики; в) участие в общественной работе предприятия (организации);

г) на какой самостоятельной работе может быть использован студент по окончании учебного заведения.

## Подпись руководителя и печать профильной

**организации**