МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Пятигорский институт (филиал) СКФУ

Колледж Пятигорского института (филиал) СКФУ

ПМ.03 Проектирование и разработка информационных систем

МДК.03.03 Сопровождение информационных систем

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

Специальности СПО

09.02.11 Разработка и управление программным обеспечением

Пятигорск 2026

Методические указания для практических занятий по дисциплине МДК.03.03 Сопровождение информационных систем составлены в соответствии с требованиями ФГОС СПО. Предназначены для студентов, обучающихся по специальности 09.02.11 Разработка и управление программным обеспечением

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

В результате освоения учебной дисциплины обучающийся должен**:**

знать:

регламенты и нормы по обновлению и техническому сопровождению обслуживаемой информационной системы; политику безопасности в современных информационных системах;

достижения мировой и отечественной информатики в области интеллектуализации информационных систем;

принципы работы экспертных систем.

уметь: осуществлять настройку информационной системы для пользователя согласно технической документации; применять основные правила и документы системы сертификации Российской Федерации; применять основные технологии экспертных систем;

разрабатывать обучающие материалы для пользователей по эксплуатации информационных систем.

иметь практический опыт в: инсталляции, настройка и сопровождение информационной системы; выполнении регламентов по обновлению, техническому сопровождению и восстановлению данных информационной системы.

ОБЩИЕ МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ

ОБЩИЕ МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ

По дисциплине Сопровождение информационных систем практических работ содержат задачи и теоретические вопросы. Варианты для каждого обучающегося - индивидуальные. Задачи и ответы на вопросы, выполненные не по своему варианту, не засчитываются.

Практическая работа выполняется студентами на персональном компьютере с использованием соответствующего программного обеспечения. В процессе выполнения задания обучающиеся выполняют практические действия, направленные на освоение изучаемых технологий и инструментов. Полученные результаты работы, а также выполненные операции и промежуточные этапы фиксируются и оформляются в виде отчёта по практической работе.

Отчёт должен содержать условия задания, описание хода выполнения работы, используемые инструменты и методы, а также полученные результаты. При необходимости в отчёт включаются схемы, таблицы, иллюстрации и другие материалы, подтверждающие выполнение задания. Все результаты должны быть представлены в структурированном виде и соответствовать требованиям оформления учебных работ.

При выполнении практической работы необходимо следовать методическим указаниям: повторить краткое содержание теории, запомнить основные формулы и законы, проанализировать пример выполнения аналогичного задания, затем преступить непосредственно к решению задачи. К зачету допускаются студенты, получившие положительные оценки по всем лабораторным работам.

Правила выполнения практических работ.

1. Студент должен прийти на практическое занятие подготовленным к выполнению работы.
2. Каждый студент после проведения работы должен представить отчет о проделанной работе с анализом полученных результатов и выводом по работе.
3. Таблицы и рисунки следует выполнять с помощью чертежных инструментов (линейки, циркуля, и.т.д.) карандашом с соблюдением ЕСКД.
4. Расчет следует проводить с точностью до двух значащих цифр.
5. Исправления проводить на обратной стороне листа. При мелких исправлениях неправильное слово (буква, число и т.п.) аккуратно зачеркивается и над ним пишут правильное пропущенное слово (букву, число и т.п.).
6. Вспомогательные расчеты можно выполнять на отдельных листах, а при необходимости на листах отчета.
7. Если студент не выполнит практическую работу или часть работы, то он выполнит ее во внеурочное время, согласованное с преподавателем.
8. Оценку по практической работе студент получает с учетом срока выполнения работы, если;

- расчеты выполнены правильно и в полном объеме;
- сделан анализ проделанной работы и вывод по результатам работы;
- студент может пояснить выполнение любого этапа работы;
- отчет выполнен в соответствии с требованиями к выполнению работы.

Практическое занятие№ 1
Системы управления производительностью приложений. Мониторинг сетевых ресурсов.

1. **Введение**
   - Определение целей и задач практической работы.
   - Краткий обзор важности мониторинга сетевых ресурсов.
2. **Теоретическая часть**
   - Основные понятия и термины: мониторинг, сетевые ресурсы, сетевой трафик, производительность сети.
   - Виды мониторинга (проактивный, реактивный).
   - Ключевые показатели эффективности (KPI) для мониторинга сетевых ресурсов.
   - Обзор популярных инструментов для мониторинга сетевых ресурсов (например, Zabbix, Nagios, SolarWinds).
3. **Практическая часть**
   - Выбор инструмента для мониторинга сетевых ресурсов.
   - Установка и настройка выбранного инструмента.
   - Настройка параметров мониторинга (сетевые устройства, сервисы, KPI).
   - Сбор и анализ данных о производительности сети.
   - Визуализация данных мониторинга.
   - Тестирование системы мониторинга на выявление проблем в сети.
   - Анализ результатов и обсуждение возможных улучшений.
4. **Дополнительные задачи**
   - Настройка уведомлений о событиях и предупреждениях.
   - Интеграция системы мониторинга с другими инструментами управления сетью.
   - Использование исторических данных для прогнозирования нагрузки на сеть.
5. **Заключение**
   - Подведение итогов практической работы.
   - Оценка эффективности выбранного инструмента для мониторинга сетевых ресурсов.

КОНТРОЛЬНЫЕ ВОПРОСЫ:

Системы управления производительностью приложений

- **Сформулировать задачи, которые решают системы управления производительностью приложений**. Например, вопросы могут включать:

  - Что такое системы мониторинга производительности серверного оборудования?
  - Какие функции выполняют системы мониторинга производительности?
  - Как системы мониторинга помогают вовремя заметить снижение производительности и определить проблемные системы в ИТ-инфраструктуре?
- **Описать инструменты мониторинга** производительности, например, «Диспетчер задач» в системах Windows, который позволяет отслеживать активность запущенных программ и служб, просматривать диаграммы использования процессора и памяти.
- **Привести примеры** систем управления производительностью приложений, например, систем, которые анализируют загрузку процессора и памяти, выявляют узкие места или приложения, отнимающие значительную часть ресурсов.

Мониторинг сетевых ресурсов

- **Определить, что такое мониторинг сетевых ресурсов**. Например, вопросы могут включать:

  - Что такое система мониторинга сети, которая выполняет постоянное наблюдение за ИТ-инфраструктурой в поисках медленных или неисправных систем, и при обнаружении сбоев сразу отправляет оповещение ИТ-специалистам для скорейшего устранения инцидента?

- Какие задачи решает система мониторинга сети: обеспечение бесперебойной и исправной работы сети, предоставление отчётов о состоянии сети и отчётов об отчётах о событиях?
- Какие инструменты используются для мониторинга сетевых ресурсов, например, сетевые утилиты для пересылки текстовых сообщений, управления общими ресурсами, диагностики сетевых подключений, поиска и обработки ошибок?

- **Привести примеры** систем мониторинга сетевых ресурсов, например, программ, которые реализуют мониторинг и диагностику локальных сетей (Monitor It, Nautilus NetRanger, CiscoWorks2000, ServiceSentinel).

Практическая работа

Некоторые задания для практической работы по темам «Системы управления производительностью приложений» и «Мониторинг сетевых ресурсов»:

- **Работа с диспетчером задач**. Например, задания могут включать:

- Запустить и принудительно завершить какое-либо приложение.
- Зайти в систему под другой учётной записью и запустить какие-либо два приложения, снова зайти в систему под своей учётной записью.
- Настроить представление запущенных процессов так, чтобы видеть имена пользователей, запустивших эти процессы.
- Принудительно отключить подключившегося пользователя или пользователя, зашедшего под другой учётной записью, и принудительно завершить процессы, запущенные другим пользователем.
- Получить сведения о файлах, связанных с какими-либо запущенными процессами.

- **Оценка загрузки основных компонентов системы** с помощью значений счётчиков, показанных в виде таблиц. Выявить узкие места или приложения (процессы), отнимающие значительную часть ресурсов компьютера.
- **Использование сетевых утилит** для диагностики сетевых подключений, поиска и обработки ошибок. Например, задания могут включать:

- Проверить корректность подключения клиента к сети, наличие у клиента хотя бы одного протокола сервера.
- Просмотреть список всех сетевых портов на компьютере и сосчитать количество открытых (прослушиваемых).
- Выполнить трассировку маршрута для удалённого узла сети, записать полученную информацию в отчёт.

- **Использование пользовательских приложений** для мониторинга и диагностики локальных сетей, например, Monitor It, Nautilus NetRanger, CiscoWorks2000, ServiceSentinel. В заданиях можно попросить разработать руководство по мониторингу сети с использованием конкретного

Практическое занятие№ 2.
Схемы и алгоритмы анализа ошибок, использование баз знаний

Цель функционального тестирования — обнаружение несоответствий между реальным поведением реализованных функций и ожидаемым поведением в соответствии со спецификацией и исходными требованиями. Функциональные тесты должны охватывать все реализованные функции с учетом наиболее вероятных типов ошибок. Тестовые сценарии, объединяющие отдельные тесты, ориентированы на проверку качества решения функциональных задач.

Функциональные тесты создаются по внешним спецификациям функций, проектной информации и по тексту на ЯП, относятся к функциональным его характеристикам и применяются на этапе комплексного тестирования и испытаний для определения полноты реализации функциональных задач и их соответствия исходным требованиям.

В задачи функционального тестирования входят:

· идентификация множества функциональных требований;

· идентификация внешних функций и построение последовательностей функций в соответствии с их использованием в ПС;- идентификация множества входных данных каждой функции и определение областей их изменения;

· построение тестовых наборов и сценариев тестирования функций;

· выявление и представление всех функциональных требований с помощью тестовых наборов и проведение тестирования ошибок в программе и при взаимодействии со средой.

Тесты, создаваемые по проектной информации, связаны со структурами данных, алгоритмами, интерфейсами между отдельными компонентами и применяются для тестирования компонентов и их интерфейсов. Основная цель — обеспечение полноты и согласованности реализованных функций и интерфейсов между ними.

Комбинированный метод «черного ящика» и «прозрачного ящика» основан на разбиении входной области функции на подобласти обнаружения ошибок. Подобласть содержит однородные элементы, которые все обрабатываются корректно либо некорректно. Для тестирования подобласти производится выполнение программы на одном из элементов этой области.

Предпосылки функционального тестирования:

· корректное оформление требований и ограничений к качеству ПО;

· корректное описание модели функционирования ПО в среде эксплуатации у заказчика;

· адекватность модели ПО заданному классу.

Под инфраструктурой процесса тестирования понимается:

· выделение объектов тестирования;

· проведение классификации ошибок для рассматриваемого класса тестируемых программ;

· подготовка тестов, их выполнение и поиск разного рода ошибок и отказов в компонентах и в системе в целом;

· служба проведения и управление процессом тестирования;

· анализ результатов тестирования.

Объекты тестирования — компоненты, группы компонентов, подсистемы и система. Для каждого из них формируется стратегия проведения тестирования. Если объект тестирования относится к «белому ящику» или «черному ящику», состав компонентов которого неизвестный, то тестирование проводится посредством ввода внего входных тестовых данных для получения выходных данных. Стратегическая цель тестирования состоит в том, чтобы убедиться, что каждый рассматриваемый входной набор данных соответствует

ожидаемым выходным выходных данным. При таком подходе к тестированию не требуется знания внутренней структуры и логики объекта тестирования.

Проектировщик тестов должен заглянуть внутрь «черного ящика» и исследовать детали процессов обработки данных, вопросы обеспечения защиты и восстановления данных, а также интерфейсы с другими программами и системами. Это способствует подготовке тестовых данных для проведения тестирования.

Для некоторых типов объектов группа тестирования не может сгенерировать представительное множество тестовых наборов, которые демонстрировали бы функциональную правильность работы компоненты при всех возможных наборах тестов.

Поэтому предпочтительным является метод «белого ящика», при котором можно использовать структуру объекта для организации тестирования по различным ветвям. Например, можно выполнить тестовые наборы, которые проходят через все операторы или все контрольные точки компонента для того, чтобы убедиться в правильности их работы.

Международный стандарт ANSI/IEEE-729-83 разделяет все ошибки в разработке программ на следующие типы.

Ошибка (error) — состояние программы, при котором выдаются неправильные результаты, причиной которых являются изъяны (flaw) в операторах программы или в технологическом процессе ее разработки, что приводит к неправильной интерпретации исходной информации, следовательно, и к неверному решению.

Дефект (fault) в программе — следствие ошибок разработчика на любом из этапов разработки, которая может содержаться в исходных или проектных спецификациях, текстах кодов программ, эксплуатационной документация и т.п. В процессе выполнения программы может быть обнаружен дефект или сбой.

Отказ (failure) — это отклонение программы от функционирования или невозможность программы выполнять функции, определенные требованиями и ограничениями, что рассматривается как событие, способствующее переходу программы в неработоспособное состояние из-за ошибок, скрытых в ней дефектов или сбоев в среде функционирования. Отказ может быть результатом следующих причин:

· ошибочная спецификация или пропущенное требование, означающее, что спецификация точно не отражает того, что предполагал пользователь;

· спецификация может содержать требование, которое невозможно выполнить на данной аппаратуре и программном обеспечении;

· проект программы может содержать ошибки (например, база данных спроектирована без средств защиты от несанкционированного доступа пользователя, а требуется защита);

· программа может быть неправильной, т.е. она выполняет несвойственный алгоритм или он реализован не полностью.

Таким образом, отказы, как правило, являются результатами одной или более ошибок в программе, а также наличия разного рода дефектов.

Ошибки на этапах процесса тестирования. Приведенные типы ошибок распределяются по этапам ЖЦ и им соответствуют такие источники их возникновения:

· непреднамеренное отклонение разработчиков от рабочих стандартов или планов реализации;

· спецификации функциональных и интерфейсных требований выполнены без соблюдения стандартов разработки, что приводит к нарушению функционирования программ;

· организации процесса разработки — несовершенная или недостаточное управление руководителем проекта ресурсами (человеческими, техническими, программными и т.д.) и вопросами тестирования и интеграции элементов проекта.

Рассмотрим процесс тестирования, исходя из рекомендаций стандарта ISO/IEC 12207, и приведем типы ошибок, которые обнаруживаются на каждом процессе ЖЦ.

Процесс разработки требований. При определении исходной концепции системы и исходных требований к системе возникают ошибки аналитиков при спецификации верхнего уровня системы и построении концептуальной модели предметной области.

Характерными ошибками этого процесса являются:

· неадекватность спецификации требований конечным пользователям;- некорректность спецификации взаимодействия ПО со средой функционирования или с пользователями;

· несоответствие требований заказчика к отдельным и общим свойствам ПО;

· некорректность описания функциональных характеристик;

· необеспеченность инструментальными средствами всех аспектов реализации требований заказчика и др.

Процесс проектирования. Ошибки при проектировании компонентов могут возникать при описании алгоритмов, логики управления, структур данных, интерфейсов, логики моделирования потоков данных, форматов ввода-вывода и др. В основе этих ошибок лежат дефекты спецификаций аналитиков и недоработки проектировщиков. К ним относятся ошибки, связанные:

· с определением интерфейса пользователя со средой;

· с описанием функций (неадекватность целей и задач компонентов, которые обнаруживаются при проверке комплекса компонентов);

· с определением процесса обработки информации и взаимодействия между процессами (результат некорректного определения взаимосвязей компонентов и процессов);

· с некорректным заданием данных и их структур при описании отдельных компонентов и ПС в целом;

· с некорректным описанием алгоритмов модулей;

· с определением условий возникновения возможных ошибок в программе;

· с нарушением принятых для проекта стандартов и технологий.

Этап кодирования. На данном этапе возникают ошибки, которые являются результатом дефектов проектирования, ошибок программистов и менеджеров в процессе разработки и отладки системы. Причиной ошибок являются:

· бесконтрольность значений входных параметров, индексов массивов, параметров циклов, выходных результатов, деления на 0 и др.;

· неправильная обработка нерегулярных ситуаций при анализе кодов возврата от вызываемых подпрограмм, функций и др.;

· нарушение стандартов кодирования (плохие комментарии, нерациональное выделение модулей и компонент и др.);

· использование одного имени для обозначения разных объектов или разных имен одного объекта, плохая мнемоника имен;- несогласованное внесение изменений в программу разными разработчиками и др.

Процесс тестирования. На этом процессе ошибки допускаются программистами и тестировщиками при выполнении технологии сборки и тестирования, выбора тестовых наборов и сценариев тестирования и др. Отказы в программном обеспечении, вызванные такого рода ошибками, должны выявляться, устраняться и не отражаться на статистике ошибок компонент и программного обеспечения в целом.

Процесс сопровождения. На процессе сопровождения обнаруживаются ошибки, причиной которых являются недоработки и дефекты эксплуатационной документации, недостаточные показатели модифицируемости и удобочитаемости, а также некомпетентность лиц, ответственных за сопровождение и/или усовершенствование ПО. В зависимости от сущности вносимых изменений на этом этапе могут возникать практически любые ошибки, аналогичные ранее перечисленным ошибкам на предыдущих этапах.

Все ошибки, которые возникают в программах, принято подразделять на следующие классы:

· логические и функциональные ошибки;

· ошибки вычислений и времени выполнения;

· ошибки ввода-вывода и манипулирования данными;

· ошибки интерфейсов;

· ошибки объема данных и др.

Логические ошибки являются причиной нарушения логики алгоритма, внутренней несогласованности переменных и операторов, а также правил программирования. Функциональные ошибки — следствие неправильно определенных функций, нарушения порядка их применения или отсутствия полноты их реализации и т.д.

Ошибки вычислений возникают по причине неточности исходных данных и реализованных формул, погрешностей методов, неправильного применения операций вычислений или операндов. Ошибки времени выполнения связаны с не обеспечением требуемой скорости обработки запросов или времени восстановления программы.

Ошибки ввода-вывода и манипулирования данными являются следствием некачественной подготовки данных для выполнения программы, сбоев при занесении их в базы данных или при выборке из нее.

Ошибки интерфейса относятся к ошибкам взаимосвязи отдельных элементов друг с другом, что проявляется при передаче данных между ними, а также при взаимодействии со средой функционирования.

Ошибки объема относятся к данным и являются следствием того, что реализованные методы доступа и размеры баз данных не удовлетворяют реальным объемам информации системы или интенсивности их обработки.

Приведенные основные классы ошибок свойственны разным типам компонентов ПО и проявляются они в программах по разному. Так, при работе с БД возникают ошибки представления и манипулирования данными, логические ошибки в задании прикладных процедур обработки данных и др. В программах вычислительного характера преобладают ошибки вычислений, а в программах управления и обработки — логические и функциональные ошибки. В ПО, которое состоит из множества разноплановых программ, реализующих разные функции, могут содержаться ошибки разных типов. Ошибки интерфейсов и нарушение объема характерны для любого типа систем.

Анализ типов ошибок в программах является необходимым условием создания планов тестирования и методов тестирования для обеспечения правильности ПО.

На современном этапе развития средств поддержки разработки ПО (CASE-технологии, объектно-ориентированные методы и средства проектирования моделей и программ) проводится такое проектирование, при котором ПО защищается от наиболее типичных ошибок и тем самым предотвращается появление программных дефектов.

Связь ошибки с отказом. Наличие ошибки в программе, как правило, приводит к отказу ПО при его функционировании. Для анализа причинно-следственных связей «ошибка отказ» выполняются следующие действия:

· идентификация изъянов в технологиях проектирования и программирования;

· взаимосвязь изъянов процесса проектирования и допускаемых человеком ошибок;

· классификация отказов, изъянов и возможных ошибок, а также дефектов на каждом этапе разработки;- сопоставление ошибок человека, допускаемых на определенном процессе разработки, и дефектов в объекте, как следствий ошибок спецификации проекта, моделей программ;

· проверка и защита от ошибок на всех этапах ЖЦ, а также обнаружение дефектов на каждом этапе разработки;

· сопоставление дефектов и отказов в ПО для разработки системы взаимосвязей и методики локализации, сбора и анализа информации об отказах и дефектах;

· разработка подходов к процессам документирования и испытания ПО.

Конечная цель причинно-следственных связей «ошибка отказ» заключается в определении методов и средств тестирования и обнаружения ошибок определенных классов, а также критериев завершения тестирования на множестве наборов данных; в определении путей совершенствования организации процесса разработки, тестирования и сопровождения ПО.

Приведем следующую классификацию типов отказов:

· аппаратный, при котором общесистемное ПО не работоспособно;

· информационный, вызванный ошибками во входных данных и передаче данных по каналам связи, а также при сбое устройств ввода (следствие аппаратных отказов);

· эргономический, вызванный ошибками оператора при его взаимодействии с машиной (этот отказ — вторичный отказ, может привести к информационному или функциональному отказам);

· программный, при наличии ошибок в компонентах и др.

Некоторые ошибки могут быть следствием недоработок при определении требований, проекта, генерации выходного кода или документации. С другой стороны, они порождаются в процессе разработки программы или при разработке интерфейсов отдельных элементов программы (нарушение порядка параметров, меньше или больше параметров и т.п.).

Источники ошибок. Ошибки могут быть порождены в процессе разработки проекта, компонентов, кода и документации. Как правило, они обнаруживаются при выполнении или сопровождении программного обеспечения в самых неожиданных и разных ее точках.

Некоторые ошибки в программе могут быть следствием недоработок при определении требований, проекта, генерации кода или документации. С другой стороны, ошибки порождаются в процессе разработки программы или интерфейсов ее элементов (например, при нарушении порядка задания параметров связи — меньше или больше, чем требуется и т.п.).

Причиной появления ошибок — непонимание требований заказчика; неточная спецификация требований в документах проекта и др. Это приводит к тому, что реализуются некоторые функции системы, которые будут работать не так, как предлагает

заказчик. В связи с этим проводится совместное обсуждение заказчиком и разработчиком некоторых деталей требований для их уточнения.

Команда разработчиков системы может также изменить синтаксис и семантику описания системы. Однако некоторые ошибки могут быть не обнаружены (например, неправильно заданы индексы или значения переменных этих операторов).

Контрольные вопросы по темам «Схемы и алгоритмы анализа ошибок», «Использование баз знаний» могут включать вопросы о процессах идентификации и анализа ошибок в информационных системах, а также о роли баз знаний в этом процессе. Ниже приведены примеры таких вопросов.

Схемы и алгоритмы анализа ошибок

- **Сформулировать задачи, которые решают алгоритмы анализа ошибок**.
- **Описать обобщённый алгоритм анализа ошибки** с помощью системы, которая собирает данные о работе ПО, состоянии операционной системы и потреблении аппаратных ресурсов. Например, вопрос может включать этапы: программный агент собирает данные, через определённые промежутки времени собранные данные передаются подсистеме локального

анализа ошибок, результатом анализа является заключение о том, соответствуют ли данные потенциальной ошибке.

- **Описать, как последовательности событий, приводящие к ошибкам, могут автоматически выявляться** на основе методов ассоциации. Автоматически созданные правила далее могут предоставляться пользователю для утверждения.
- **Указать, как затруднён автоматический анализ сообщений от пользователей** — пользователь может дать неправильное или неполное описание ошибки, а отчёты об ошибках, выраженные в текстовом виде, сложно классифицировать. Для решения этой проблемы можно использовать методы интеллектуального анализа текстов, например, алгоритмы извлечения информации.
- **Описать, как интерфейс пользователей может предоставлять поддержку в процессе поиска первичных ошибок** — например, отображать результаты в виде специальных диаграмм, например, «деревьев ошибок».

Практическое занятие№ 3

Отчет об ошибках системы: содержание, использование информации.

Понятие отчета об ошибках Отчёт об ошибке (англ. error report или crash report) – это файл, содержащийтехническую информацию об исключительной ситуации (исключении), произошедшей в программе на компьютере пользователя. Отчеты об ошибках создаются, когда в системе возникают неполадки с аппаратным или программным обеспечением. Отчеты об ошибках содержат следующие разделы: сведения о состоянии компьютера при возникновении ошибки, версия используемой операционной системы и аппаратного обеспечения, а также цифровойкод продукта, используемый для определения лицензии. Также передается IP-адрес компьютера, поскольку для отправки отчета необходимо подключиться к сетевойслужбе.

Отчеты об ошибках могут содержать данные файлов журналов, например, имена пользователей, IP-адреса, URL-адреса, имена файлов и пути к ним, а также адреса электронной почты. Отчеты об ошибках отправляются с использованиемшифрования в базу данных с ограниченным доступом и не используются в каких- либо коммерческих целях. Настройка параметров отбора событий Можно настроить параметры сбора данных диагностики для ведения журнала событий.

События можно регистрировать как в журнале событий Windows, так ижурнале трассировки. Можно настроить параметры регулирования событий, чтобызадавать число событий, регистрируемых в каждом журнале в соответствии с критичностью события.

Для расширения возможностей управления при регулированиисобытий можно задать регулирование всех событий или любой отдельной категории событий. Доступно несколько категорий событий в зависимости от служб ивозможностей. Категории событий могут определяться по отдельным службам или по группам связанных событий. К категориям выбранных событий относятся: ⌡ - для всех; ⌡ - категории, определенные в соответствии с используемым продуктом (например, Office SharePoint Server 2007 или Microsoft Office Project Server 2007); ⌡ - административные функции, такие как «Администрирование», «Резервное копирование и восстановление», и др.

Просмотр событий - компонент, включённый в состав операционных систем, который позволяет администраторам просматривать лог событий на локальном компьютере или на удалённой машине. Записи журнала событий хранятся в файлах журналов:

⌡ файл журнала приложений — для событий приложений и служб;

⌡ файл журнала безопасности — для событий системы аудита;

⌡ файл системного журнала — для событий драйверов устройств. Для выбранной категории устанавливается минимальный уровень критичности событий, которые следует регистрировать в журнале событий Windows и в журнале трассировки. В каждом журнале будут регистрироваться события этогоили более высокого уровня критичности. Записи в этом списке сортируются

в порядке от максимального уровня критичности к минимальному. События журнала Windows могут иметь следующие уровни:

¬ - не используется; ¬ - error (ошибка);

¬ - warning (предупреждение);

¬ - не удалось выполнить аудит;

¬ - успешное выполнение аудита;

¬ - information (информация).

События журнала трассировки могут иметь следующие уровни:

¬ - не используется;

¬ - unexpected (непредвиденный);

¬ - monitorable (контролируемый);

¬ - high (высокая);

¬ - medium (средняя);

¬ - verbose (подробный).

Контрольные вопросы

1. Какие типы журналов можно просматривать средствами утилиты просмотра событий? Для чего предназначен каждый из них?
2. Какие уровни событий предусмотрены в журнале?
3. Какова структура отчета об ошибках?
4. Что такое отчет об ошибках?
5. Каковы источники информации для создания отчета об ошибках?
6. Какая информация содержится в протоколе ошибок?
7. Как формируется протокол ошибок?
8. Как можно очистить протокол ошибок?
9. В каких случаях очистка проводится автоматически?
10. Какие возможности имеются у администратора системы?

Практическое занятие№ 4

Методы и инструменты тестирования приложений. Пользовательская документация: «Руководство программиста», «Руководство»

Что такое руководство программиста

Руководство пользователя (РП) – это документ, который помогает в использовании продукта, устройства или услуги. Руководства пользователя обычно содержат пошаговые инструкции, иллюстрации, аннотации, часто задаваемые вопросы и информацию по устранению неполадок.

Руководство программиста отличается от руководства пользователя тем, что оно предназначено для аудитории со знанием языков программирования, особенностей ПО и сред разработок. Его текст написан формальным и точным языком, содержит технические термины, предполагая наличие у читателя базовых знаний в области программирования. РП фокусируется на практических аспектах использования инструмента или языка, включая синтаксис, функции и примеры кода.

Функции руководства программиста

Руководство программиста выполняет следующие функции:

- **Пошаговые инструкции**. Данная документация обеспечивает подробные инструкции по использованию конкретного инструмента или языка программирования, что помогает разработчикам быстро приступить к работе.
- **Примеры кода**. Многие РП включают пример кода, которые показывают, как его использовать на практике. Эти примеры помогают разработчикам избежать распространенных ошибок.
- **Справочная информация**. Руководства программиста служат справочным материалом, к которому разработчики могут обращаться при необходимости.
- **Сокращение времени разработки**. Использование руководства может значительно сократить время разработки. Разработчики могут быстро найти необходимую информацию и приступить к написанию кода, не тратя время на изучение документации или поиск ответов и сообщения на форумах.
- **Повышение качества кода**. Руководства программиста помогают разработчикам писать более качественный код, предоставляя рекомендации по структурированию кода и указания, как избежать ошибок.

Таким образом, **руководство программиста является важным документом, который обеспечивает разработчикам всеобъемлющую справочную информацию и помощь в разработке**. Данный ресурс помогает сократить время, улучшить качество кода и в целом делает рабочий процесс более продуктивным.

Требования к содержанию и оформлению руководства программиста устанавливает **ГОСТ 19.504-79** – государственный стандарт, содержащий перечень требований к документации программы, включая руководства программиста. Согласно этому документу, руководство программиста должно содержать следующую информацию:

- наименование и назначение программы,
- её технические характеристики,
- а также требования к функционированию.

Задачи руководства программиста

Ниже приведены основные задачи, которые выполняет руководство программиста:

- **Объяснение интерфейсов и функций**. Описание методов, классов, параметров и других элементов интерфейса.
- **Предоставление примеров кода**. Включение примеров кода, демонстрирующих использование интерфейсов и функций, а также лучших практик программирования.
- **Устранение неполадок и диагностика**. Описание возможных ошибок и проблем, с которыми могут столкнуться программисты, а также способы их устранения.
- **Рекомендации и ограничения**. Указание рекомендаций по эффективному использованию программного обеспечения и предупреждение о любых ограничениях или проблемах совместимости.

- **Справочная документация**. Предоставление справочных материалов по конкретным функциям, типам данных и другим техническим аспектам.

Хорошо написанное руководство программиста является ценным ресурсом, позволяющим быстро начать разработку, понять сложные технические детали, а также избежать распространенных ошибок и подводных камней.

Структура руководства программиста

При создании руководства для программиста эффективная структура имеет решающее значение для обеспечения удобства использования, доступности и полноты информации.

Руководство программиста должно содержать следующие разделы:

- **Титульный лист**. В этом разделе нужно упомянуть название организации, разработавшей руководство; наименование, идентификационный номер и версию программы, а также дату введения документа в действие.
- **Введение**. Раздел кратко описывает назначение программы, область её применения, основные функции и ограничения.
- **Назначение и область применения программы.** Здесь должны быть указаны функции, выполняемые программой, условия, необходимые для выполнения программы (объем оперативной памяти, требования к программному обеспечению и т.п.).
- **Характеристика программы.** Данная секция должна включать в себя описание основных особенностей программы (время выполнения различных задач, режим работы и т.п.).
- **Требования к функционированию программы.** Этот раздел описывает организацию используемой входной и выходной информации, а также её кодирования.
- **Приложение.** В приложении для руководства программиста может быть приведена дополнительная информация (примеры, иллюстрации, графики, таблицы и т.д.).
- **Контактная информация.** При возникновении вопросов или проблем пользователи смогут связаться с авторами.

Создание руководства программиста в соответствие со структурой, приведенной выше, может значительно облегчить процесс его написания. При необходимости может быть добавлен один или несколько разделов, например «Лицензия» и т.п.

Виды руководства программиста

Хотя общие принципы руководства программиста остаются прежними, существуют различные виды руководств, которые определяются такими факторами как требования проекта, язык программирования, состав команды разработчиков, а также инструменты и технологии, используемые в работе. Более подробно данные факторы описаны ниже:

- **Конкретные требования проекта.** Крупные проекты могут требовать более строгих руководств, в то время как для небольших проектов может потребоваться более гибкий подход. Руководства программиста должны учитывать специфические потребности и ограничения бизнес-домена. Например, финансовые приложения могут потребовать более высоких стандартов безопасности, а приложения пользовательского интерфейса могут уделять больше внимания удобству использования.
- **Язык программирования.** Руководства программиста должны соответствовать правилам синтаксиса конкретного языка программирования. Например, некоторые языки используют отступы для обозначения блоков кода, в то время как другие используют фигурные скобки. Также различные языки программирования имеют разные механизмы управления памятью, что требует соответствующих руководств по использованию памяти.
- **Состав команды разработчиков.** Навыки и опыт членов команды разработчиков должны учитываться при разработке руководства программиста. Более опытным командам может потребоваться менее подробное руководство, в то время как менее опытным командам может потребоваться более подробный и прямой подход.
- **Технологии**. Инструменты, используемые командой, также могут влиять на выбор руководства программиста. Например, использование системы контроля версий может потребовать руководство по управлению ветками и разрешению конфликтов слияния.

Вариации руководства программиста отражают уникальные потребности и обстоятельства различных проектов, языков программирования и команд разработчиков.

Понимая и учитывая эти вариации, можно создать руководства программиста, которые эффективно способствуют реализации высококачественного, согласованного кода.

СДК как вариация руководства программиста

СДК (от англ. SDK — software development kit, рус. «комплект для разработки ПО»), также известный как инструментарий разработки программного обеспечения, представляет собой набор инструментов, библиотек и документации, которые предоставляют разработчикам необходимые ресурсы для создания приложений на определенной платформе или с использованием определенных технологий.

SDK включают в себя следующие компоненты: **документация и справочные материалы, примеры кода, библиотеки кода, инструменты отладки и тестирования, интерфейсы программирования приложений (API)** для взаимодействия с платформой или сервисом.

SDK и руководства для разработчиков имеют много общего, но есть и существенные различия. **Руководства для разработчиков в первую очередь ориентированы на предоставление информации**, тогда как SDK также предоставляют инструменты и ресурсы для практического применения. SDK часто включают инструменты, позволяющие программистам интерактивно взаимодействовать с платформой или сервисом, например среды разработки и отладчики. Помимо этого SDK предназначены для бесшовной интеграции с существующими проектами разработки, предоставляя готовые к использованию компоненты и расширяя возможности разработки.

SDK стали **неотъемлемой частью современного набора инструментов разработчика**, расширяя функциональность руководства для разработчиков. Предоставляя инструменты, ресурсы и расширенные возможности, SDK способствуют повышению эффективности, качества и безопасности разработки программного обеспечения. При выборе и использовании SDK разработчики должны учитывать особенности конкретных платформ, сервисов и приоритетов разработки, чтобы найти наилучшее решение для своего проекта.

Инструменты для написания руководства программиста

Написание руководств — сложная и трудоемкая задача. Чтобы создать высококачественную документацию необходимо учитывать множество факторов, таких как ясность, точность и полнота. К счастью, существует ряд инструментов, которые могут помочь в этой задаче.

Одним из таких ресурсов является облачная платформа **Документерра**. Это популярный инструмент для написания руководства программиста, располагающий рядом преимуществ, которые упрощают создание и управление документацией, в том числе:

- **Централизованное хранение и управление документацией.** Документерра предоставляет централизованное хранилище для всех руководств, что упрощает доступ и управление информационными ресурсами проекта. В системе можно легко организовывать руководства по категориям, темам или продуктам, что облегчит навигацию для пользователей. Платформа позволяет управлять версиями, поэтому в Документерре можно отслеживать изменения и откатываться к предыдущим версиям, если это необходимо.
- **Доступность на различных устройствах.** Руководства программистов, созданные на Документерре, доступны на всех устройствах и платформах, включая настольные компьютеры, мобильные устройства и планшеты. Пользователи могут получить доступ к руководствам в любое время и в любом месте, что делает документацию доступной для специалистов, работающих в удаленной среде.
- **Возможность совместной работы.** Команды разработчиков могут одновременно вносить изменения, оставлять комментарии и отслеживать ход работы.
- **Аналитика и отчетность.** Платформа предоставляет аналитические данные и отчеты об использовании руководств, что позволяет пользователям отслеживать их эффективность. В системе есть возможность мониторинга таких параметров как популярность руководств и частота просмотров. На этой основе можно делать выводы о необходимости улучшений.

- **Настраиваемый интерфейс и брендинг.** Документерра позволяет настраивать внешний вид и оформление руководств, чтобы они соответствовали бренду и стилю компании. Система позволяет загружать собственные логотипы, цветовые схемы и шрифты.

Документерра – это удобная платформа для создания и управления РП, предлагающая ряд преимуществ. Централизованное хранение, интерактивные функции, интеграция с системами контроля версий и возможность совместной работы делают Документерру идеальным решением для разработки высококачественной технической документации.

* * *

Руководство программиста – это незаменимый инструмент для любого программиста, независимо от его уровня подготовки. Эффективное руководство способствует четкому пониманию и реализации проекта, а также облегчает устранение неполадок. Кроме того, качественное руководство программиста повышает ценность приложения, сайта или программного обеспечения, делая его более доступным и простым в использовании и обслуживании.

Таким образом, РП является неотъемлемой частью любого успешного проекта и должно рассматриваться как ценный ресурс для всех, кто участвует в его разработке и внедрении.

Практическое занятие№ 5

Выявление аппаратных ошибок информационной системы. Техническое обслуживание аппаратных средств.

Выявление аппаратных ошибок

Некоторые методы выявления аппаратных ошибок информационной системы:

- **Тестирование компонентов системы**. Проведение специализированных диагностических тестов.
- **Использование резервирования**. Применение аппаратной, программной, информационной или временной избыточности системы для сравнения данных с их копиями или результатов нескольких вычислений с целью выявления расхождений.
- **Реверсивная проверка**. На основе значения, полученного на выходе, делается предположение о значении на входе, после чего эти значения сравниваются. Выявляется несоответствие рассчитанного значения входа и его текущего значения.
- **Кодирование**. Проверяется соответствие текущих значений кодов для данных и команд с их заранее вычисленными эталонными значениями. Выявляется их несоответствие.
- **Проверка функциональной корректности**. Используются знания о системе для проверки корректности значений некоторых величин.

Также для выявления аппаратных отказов используют **модули контроля**, которые определяют технические неисправности компонентов. Отказы и сбои определяются путём сравнения сигналов на входах и выходах и параметрической регуляции модуля. При отклонении модуля от установленного порядка фиксируется неисправность, после чего информация передаётся рабочей станции.

Для диагностики аппаратных неполадок также применяют **отчёты об ошибках** — файлы, содержащие техническую информацию об исключительной ситуации в системе. В отчётах указывают сведения о состоянии компьютера при возникновении ошибки, версию используемой операционной системы и аппаратного обеспечения и другие данные.

Техническое обслуживание

Некоторые принципы технического обслуживания аппаратных средств информационных систем:

- **Выполнение регламентных работ** в соответствии с технической документацией производителя.
- **Осуществление обслуживания** должно осуществляться подразделением или лицом, ответственным за техническое обслуживание средств вычислительной техники, а также привлекаемыми специалистами (при гарантийном обслуживании).
- **Выполнение мер по защите информации**. Например, если работы проводятся сторонней организацией за пределами контролируемой зоны, защищаемая информация должна быть удалена с передаваемых носителей информации.
- **Соблюдение требований поставщика технических средств** для выполнения гарантийных обязательств.

При выполнении обслуживания системы рекомендуется заносить в системный протокол ошибок информацию о выполненных действиях, например, об очистке протокола ошибок, замене аппаратного компонента или применении исправления. t

Важно своевременно проводить диагностику аппаратных средств, чтобы избежать их некорректной работы и поломок.

Проверка промышленного оборудования

Для повышения квалификации технических специалистов можно использовать учебный стенд **«Балтех — BALTECH»**. Он предназначен для специалистов ремонтных служб промышленных предприятий, слушателей вузов, колледжей и учебных комбинатов. С его помощью можно проводить **лазерную диагностику**, анализировать геометрические параметры турбин, компрессоров, станков, а также измерять вибрацию и вибродиагностику для определения состояния оборудования.

Практическое занятие№ 6.

Типовая система технического профилактического обслуживания и ремонта

Типовая система технического профилактического обслуживания и ремонта (ТОиР) — это совокупность взаимосвязанных средств, документации и исполнителей, необходимых для поддержания и восстановления работоспособности объектов, входящих в эту систему.

Цель ТОиР — поддержание работоспособности оборудования, увеличение срока его эксплуатации и уменьшение вероятности отказа. Для достижения этой цели специальное подразделение или внешний подрядчик на системном уровне решает задачи: обеспечивает регулярные осмотры и работы по обслуживанию и ремонту оборудования, организует анализ технического состояния оборудования и затрат на его обслуживание и ремонт, контролирует безопасность проведения этих работ.

Виды

Некоторые виды ТОиР:

- **Регламентированное техническое обслуживание** — плановое обслуживание, выполняемое с установленной в документации периодичностью независимо от состояния объекта на момент начала обслуживания. К регламентированным работам относятся, например, мойка, чистка и регулировка узлов и агрегатов, замена смазки и расходников, испытания на прочность.
- **Нерегламентированное техническое обслуживание** — обслуживание, обусловленное особыми условиями эксплуатации или ненормированной наработкой объекта и его составных частей. Например, ремонт отдельных деталей или сборочных единиц, неисправность которых обнаруживается только при разборке объекта.
- **Профилактическое техническое обслуживание** — плановое обслуживание, выполняемое через определённые интервалы времени и направленное на поддержание работоспособного состояния объекта, на раннее выявление неисправностей и снижение вероятности отказов.
- **Корректирующее техническое обслуживание** — обслуживание, выполняемое после обнаружения неисправности с целью возвращения объекта в работоспособное состояние.

Принципы

Некоторые принципы организации ТОиР:

- **Единство основания для выполнения работы** — каждая работа должна быть связана с конкретным видом отказа и направлена на его предупреждение или устранение.
- **Учёт периодичности обслуживания** — периодичность может прописывать производитель в технической документации, опираясь на часы наработки, пробеги устройства или временные интервалы.
- **Учёт условий эксплуатации** — например, сезонное техническое обслуживание для подготовки объекта к использованию в осенне-зимних или весенне-летних условиях.
- **Автоматизация процессов** — использование специализированных программных решений для автоматизации управления ТОиР, которые помогают планировать и контролировать обслуживание, учитывать историю ремонтов, управлять запасами и анализировать техническое состояние активов.

Документация

В системе ТОиР могут использоваться:

- **Директивные документы** — «Положение о ТОиР производственного оборудования», «Правила технической эксплуатации оборудования», «Инструкции по

техническому обслуживанию оборудования». Первые два документа носят отраслевой характер, инструкции обычно утверждаются руководством предприятия.

- **Технологическая документация** — документы по технологическим процессам ремонта (в том числе маршрутные, операционные и технологические карты, технологические инструкции, рабочие программы).
- **Организационно-распорядительная документация** — документы по планированию, подготовке и выполнению ТОиР, а также учёта и отчётности (планы, графики, программы ремонта, ведомости, протоколы, акты).

Нормативные документы

Деятельность, связанная с ТОиР, регулируется, например:

- **ГОСТ 18322-2016** — межгосударственный стандарт «Система технического обслуживания и ремонта техники. Термины и определения».
- **Локальные нормативные акты** — например, «Правила организации технического обслуживания и ремонта объектов электроэнергетики», утверждённые приказом Министерства энергетики РФ от 25.10.2017 №1013. В этих актах регламентируются организация ТОиР оборудования, зданий и сооружений, порядок и правила взаимодействия лиц, осуществляющих ремонтную деятельность

Контрольные вопросы

- **Какие виды технического обслуживания** существуют в типовой системе? Например:
  - **Периодическое** — выполняется через установленные в документации значения наработки или интервалы времени.
  - **Регламентированное** — предусмотрено в нормативно-технической и эксплуатационной документации, выполняется с периодичностью и в объёме, установленными в ней, независимо от технического состояния объекта в момент начала обслуживания.
  - **Сезонное** — осуществляется для подготовки объекта к использованию в осенне-зимних или весенне-летних условиях.
- **Какие особенности технического обслуживания** (например, часть работ выполняется только во время работы объекта — «на ходу»)?
- **Какие цели профилактических осмотров** в типовой системе? Например, выявление неисправностей, способных вызвать поломку, или установление технического состояния наиболее ответственных деталей и узлов для уточнения объёмов предстоящих плановых ремонтов.

Основная литература:

1. Яхъяева, Г.Э. Основы теории нейронных сетей / Г.Э. Яхъяева. - 2-е изд., испр. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 200 с. : ил. - (Основы информационных технологий). - ISBN 978-5-94774-818-5 ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=429110
2. Лягинова, О.Ю. Разработка схем и диаграмм в Microsoft Visio 2010 / О.Ю. Лягинова. - 2-е изд., исправ. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 128 с. : схем., ил. ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=428810
3. Кияев, В.И. Развитие информационных технологий / В.И. Кияев, О.Н. Граничин. - 2-е изд., исправ. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 199 с. : схем., ил. - Библиогр. в кн. ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=428804

Дополнительная литература:

1. Фуфаев Э.В. Разработка и эксплуатация удаленных баз данных: учебник для студ. учреждений сред.проф. образования/ Э.В.Фуфаев, Д.Э. Фуфаев. – 4-е изд., стер. – М.: Издательский центр «Академия», 2014. – 256 с.
2. Волкова, Т.В. Основы проектирования компонентов автоматизированных систем: учебное пособие / Т.В. Волкова ; Министерство образования и науки Российской Федерации, Оренбургский Государственный Университет, Кафедра программного обеспечения вычислительной техники и автоматизированных систем. - Оренбург : ОГУ, 2016. - 226 с. : табл., ил. - Библиогр. в кн. - ISBN 978-5-7410-1560-5 ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=471129
3. Леоненков, А. Визуальное моделирование в среде IBM Rational Rose 2003 / А. Леоненков. - 2-е изд., исправ. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 193 с. : ил. - (Основы информационных технологий). - Библиогр. в кн. ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=429149