Слайд 2 Введение

Инспекция исходного текста - это процедура проверки качества исходного кода программы. Она позволяет выявить и исправить ошибки и недочёты ещё на предварительном этапе разработки, что в свою очередь снижает количество ошибок в готовом продукте. Данная процедура включает в себя контроль обращений к данным, контроль вычислений, контроль передачи управления и контроль межмодульных интерфейсов. Общая процедура проведения инспекции исходного текста включает несколько этапов, таких как:

1. подготовка к инспекции;
2. инспекция;
3. анализ результатов инспекции;
4. исправление найденных ошибок и недочётов.

Контроль обращений к данным включает в себя проверку корректности доступа к данным (их чтение, запись, изменение) и проверку их соответствия типу переменных.

Контроль вычислений позволяет определить правильность математических вычислений и использование арифметических операций в соответствии с их приоритетом.

Контроль передачи управления включает проверку правильности использования условных операторов и циклов, а также проверку правильности логических операций.

Контроль межмодульных интерфейсов включает проверку правильности связей между различными модулями программы, корректность передачи параметров между ними, а также соответствие их типов.

В целом, проведение инспекции исходного текста является важным этапом в процессе разработки программного обеспечения, который помогает обеспечить высокое качество кода и повышает эффективность работы всей команды.

СЛАЙД 3 Цель инспекции исходного текста

Цель инспекции исходного текста состоит в том, чтобы найти и исправить ошибки и недочёты в коде программы ещё на ранних стадиях разработки, когда их исправление значительно дешевле и проще, чем в готовом продукте. Также целями инспекции являются повышение качества программы, уменьшение рисков возникновения дефектов и сокращение времени, затрачиваемого на доработку кода после того, как она уже была написана.

Другие цели, которые можно достичь благодаря проведению инспекции, включают:

снижение вероятности появления ошибок в программе, которые могут привести к сбоям или отказам;

повышение понимания кода всеми участниками проекта, что позволяет ускорить процесс разработки и уменьшить время на решение проблем;

улучшение коммуникации и сотрудничества между членами команды исходной разработки.

В общем, цель инспекции исходного текста заключается в том, чтобы обеспечить качественную разработку программного обеспечения путём обнаружения и устранения ошибок и недостатков в начальном этапе.

СЛАЙД 4 Приложения для Инспекции исходного текста.

Для инспекции исходного текста в js используются Линтеры.

Линтеры (от англ. Linter) - это инструменты, которые используются для анализа и проверки исходного кода на наличие ошибок, недочетов и нарушений принятых кодировочных правил.

В общем, линтеры нужны для того, чтобы помочь разработчикам писать более правильный, эффективный и безопасный код. Они могут проверять синтаксис, структуру, форматирование, объявление переменных, использование модулей и другие аспекты кода.

Примеры линтеров для JavaScript: ESLint, JSLint, JSHint. Они позволяют настроить правила проверки под конкретные потребности и предоставляют пользователю информацию о найденных проблемах и способы их решения.

СЛАЙД 5 Приложения JSLint

JSLint - это инструмент для статического анализа JavaScript-кода, разработанный Дугласом Крокфордом. JSLint проверяет исходный код на соответствие определенной многими считаемой за идеальную чистоте и стандартам языка JavaScript. JSLint может быть полезен для выявления ошибок в коде, а также для облегчения чтения и понимания написанного кода.

JSLint имеет свою версию и в виде онлайн-инструмента на официальном сайте <http://jslint.com/>, и в виде небольших плагинов для различных редакторов кода, например, для Sublime Text, Visual Studio или Visual Studio Code.

Однако, учитывая, что стандарты, по которым JSLint проверяет код, достаточно жесткие, он может выдавать много ошибок и предупреждений, которые на самом деле не являются ошибками, но могут показаться таковыми.

СЛАЙД 6 Этапы проведения инспекции исходного текста

Этапы проведения инспекции исходного текста могут различаться в зависимости от методологии инспекции и конкретных инструментов, используемых при проведении. Но общая схема такая:

1. Подготовка к инспекции:

формулируются цели инспекции (например, выявление ошибок, повышение качества кода);

определяются стандарты, соглашения или правила, которым должен соответствовать код;

выбирается инструментарий для инспекции (например, линтеры).

1. Настройка инструментов:

устанавливаются и настраиваются инструменты на соответствие используемым стандартам и правилам.

1. Проведение инспекции:

код проходит автоматические процедуры анализа с помощью подготовленных инструментов;

определяются зоны ответственности и отдельные участки кода, которые будут инспектироваться более подробно;

проводится ручное исследование кода с целью выявления ошибок, недочетов и нарушений правил.

1. Оформление результатов:

выявленные ошибки фиксируются и документируются (либо в самом коде, либо в отдельном отчете);

рекомендации по устранению ошибок и улучшению кода формулируются и документируются.

1. Устранение ошибок и улучшение кода:

выявленные ошибки исправляются;

для улучшения кода внедряются рекомендации и советы, полученные в результате инспекции.

1. Итоговое тестирование:

код проверяется на соответствие критериям качества, определенным на этапе подготовки;

при необходимости, повторяются этапы проведения инспекции и улучшения кода.

СЛАЙД 7 Описание общей процедуры проведения инспекции

Общая процедура проведения инспекции включает следующие этапы:

1. Подготовка

На первом этапе производится формирование целей и задач инспекции, выбирается методология и инструменты анализа. Также на этом этапе определяются стандарты качества, правила написания кода, кодирование и кодирование для проекта, которые будут использоваться при анализе.

1. Проверка автоматических инструментов

На этом этапе производится запуск автоматических инструментов, таких как линтеры, анализаторы статического кода, системы контроля версий, тестовые средства, которые помогают выявить нарушения кода.

Анализ кода на уровне переменных и функций

1. На этом этапе производится анализ кода на уровне переменных и функций, выявление зон риска. Это инспектирование происходит вручную, что требует много времени и усталости наблюдателя.

Оценка стилевых конвенций

1. На этом этапе анализируются всевозможные правила и стандарты написания кода, включая кодировку, определение типов данных, идентификацию объектов и конвенции в имени, размер функций и классов, а также комментарии и документация.
2. Определение пропущенного функционала

Здесь производится выявление расхождений между требованиями проекта и реализацией в процедурном коде, посредством которой может быть выявлен функционал, который отсутствует или работает неправильно.

1. Превентивные меры

Окончательный шаг состоит в предоставлении рекомендаций по превентивным мерам, созданию чеклиста и комментариям авторам кода. Эти меры проектируются с целью устранения недостатков, определения вопросов безопасности и усовершенствования сценария. Результатом инспекции кода должна стать документация, где будут выделены все недостатки, найденные в коде, а также рекомендации по их устранению.

СЛАЙД 8 Контроль обращений к данным

Контроль обращений к данным означает установление системы ограничений и правил доступа к данным в информационной системе.

Это может включать в себя контроль доступа к данным на уровне пользователя, группы пользователей, роли пользователя, уровня допуска и других факторов. Контроль обращений к данным позволяет обеспечить безопасность данных и предотвратить несанкционированный доступ к чувствительной информации.

СЛАЙД 9 Контроль вычислений

Контроль вычислений — это процесс проверки корректности результатов выполнения вычислительных операций с использованием программного обеспечения (ПО) и аппаратного обеспечения (АО).

Он включает в себя:

1. проверку правильности входных данных;
2. проверку методов вычислений;
3. проверку используемого программного и аппаратного обеспечения
4. проверка результатов вычислений.

Цель контроля вычислений заключается в том, чтобы обеспечить точность результатов и убедиться, что результаты вычислений удовлетворяют заданным ограничениям и правилам.

СЛАЙД 10 Контроль передачи управления

Контроль передачи управления - это процедура, при которой программа убеждается в правильном и безопасном передачи управления от одной части программы к другой.

Он включает в себя:

1. проверку правильности условных операторов и переходов на метки;
2. проверку наличия необходимых переменных и выражений;
3. подтверждение, что передача управления не приведет к ошибкам, неожиданному поведению программы или ее остановке.

Контроль передачи управления является важным аспектом программирования, поскольку если передача управления выполняется неправильно, это может привести к ошибкам, сбоям программы или даже к безопасности компьютерной системы в целом.

Многие современные языки программирования предоставляют системные средства контроля передачи управления, такие как проверка границ массива и обработка исключений, которые позволяют программистам более эффективно контролировать передачу управления и обеспечить безопасность программного кода.

СЛАЙД 11 Описание методов контроля передачи управления

Существуют различные методы контроля передачи управления в программах, которые программисты используют для обеспечения правильности и безопасности своих программ. Рассмотрим некоторые из них:

1. Проверка наличия необходимых переменных и выражений перед передачей управления. Это включает проверку наличия данных, которые используются в условном операторе или в переходе на метку. Если такие данные необходимы, но отсутствуют, программа не будет выполняться правильно.
2. Проверка наличия границ в массивах. Этот метод контроля передачи управления используется для обеспечения безопасности памяти и предотвращения ошибок, связанных с обращением к несуществующим элементам массива или выходом за его границы.
3. Обработка исключений. Этот метод контроля передачи управления используется для обработки ошибок, которые могут возникнуть в процессе выполнения программы. Он позволяет программистам определить, как следует обрабатывать ошибки, и принимать соответствующие меры.
4. Использование отладочных инструментов. Отладчик - это инструмент, который используется при разработке программы для поиска и исправления ошибок. Он позволяет программеру проводить отладку своей программы, шаг за шагом, и контролировать передачу управления в процессе выполнения.
5. Тестирование программы. Этот метод контроля передачи управления заключается в том, чтобы написать набор тестов для проверки поведения программы в разных условиях. Тестирование позволяет программисту убедиться в том, что программа выполняет свои функции правильно и безопасно.

СЛАЙД 12 Примеры ошибок, связанных с передачей управления

Некоторые примеры ошибок, связанных с передачей управления в программировании:

1. Ошибка "segmentation fault" (Ошибка сегментирования) - возникает, когда программа пытается обратиться к памяти, которой не существует или к которой нет доступа. Это может произойти, если произойдет выход за границы массива или при обращении к переменной, которая не была инициализирована.
2. Бесконечный цикл - возникает, когда программа застревает в цикле, и не может выполниться дальше. Это может произойти, если условие выхода из цикла никогда не становится истинным.
3. Неверная передача аргументов функции - возникает, когда аргументы передаются в функцию в неправильном порядке или с неправильным типом данных.
4. Неправильная обработка исключений - возникает, если программа не справляется с ошибками, возникающими в процессе выполнения исключений, и продолжает выполнение в неправильном состоянии.
5. Неправильное использование команды return - возникает, когда команда return используется в неправильном месте в коде, что может привести к непредсказуемому поведению программы.

СЛАЙД 13 Контроль межмодульных интерфейсов

Контроль межмодульных интерфейсов является важным этапом разработки программного обеспечения. Это позволяет обеспечить правильную и соответствующую работу программы в целом и ее отдельных модулей в частности.

Основными методами контроля межмодульных интерфейсов являются:

1. Декомпозиция системы на части или модули. Каждый модуль должен быть независимым и выполнять четко определенную функцию.
2. Описать интерфейсы между модулями. Это позволяет явно определить, какие данные могут передаваться между модулями и какие функции могут вызываться.
3. Проверка типов данных и структуры данных. Это помогает гарантировать, что данные, передаваемые между модулями, правильные и соответствуют ожиданиям.
4. Тестирование. Тестирование модулей и их взаимодействия помогает выявить ошибки в работе модулей и их интерфейсов, а также находить ошибки при работе системы в целом.
5. Общение с другими разработчиками. Разработчики должны взаимодействовать и общаться друг с другом, чтобы убедиться, что интерфейсы между модулями были правильно разработаны и правильно используются.

СЛАЙД 14 Описание методов контроля межмодульных интерфейсов

Методы контроля межмодульных интерфейсов в программировании имеют целью обеспечить правильную работу программных модулей и находящихся между ними связей.

Основные методы контроля межмодульных интерфейсов:

1. Декомпозиция системы на подсистемы или модули

Это позволяет разбить сложную систему на более мелкие модули. Каждый модуль должен выполнять четко определенную функцию и быть независимым от других модулей.

1. Проработка интерфейсов между модулями

При проектировании интерфейсов необходимо определить, какие данные и команды могут передаваться между модулями. Проработка и документирование интерфейсов позволяет предотвратить несовпадение ожиданий взаимодействия модулей.

1. Проверка типов данных

При передаче данных между модулями необходимо проверить, что типы данных совпадают. Также необходимо проверить корректность структуры передаваемых данных.

1. Тестирование интерфейсов

Тестирование модулей и их взаимодействия помогает проверить корректность работы интерфейсов между модулями. Тестирование может проводиться вручную или автоматически.

1. Использование определенных стандартов

Использование определенных стандартов и контрольных точек при проектировании интерфейсов позволяет удостовериться в совместимости модулей, использовать общие соглашения и улучшить эффективность работы.

1. Резервное копирование

Создание резервной копии данных, передаваемых между модулями, является общепринятой практикой. Это позволяет быстро восстановить функциональность системы в случае необходимости. Контроль межмодульных интерфейсов в программировании является очень важным этапом, который позволяет обеспечить правильную работу и координацию между различными частями программного продукта.

СЛАЙД 15 Заключение

Контроль межмодульных интерфейсов в программировании является важным этапом проектирования и разработки программного обеспечения. Правильное и эффективное определение интерфейсов между модулями и контроль их работы позволяет создавать системы, работающие надежно, эффективно и без сбоев.

Без контроля межмодульных интерфейсов не только возможны проблемы в работе программной системы, но и увеличивается сложность ее дальнейшей поддержки и обновления.

Поэтому необходимо уделить большое внимание контролю и проработке межмодульных интерфейсов при разработке программного обеспечения, чтобы обеспечить его успешную реализацию и использование.