**Динамические и статические методы при структурном и объектном подходах тестирования.**

**Динамические методы тестирования**

Динамическое тестирование включает выполнение кода программы для проверки его работы и выявления дефектов.

**Структурный подход:**

* Покрытие операторов: Убедиться, что каждый оператор в коде выполняется хотя бы один раз.
* Покрытие ветвей: Проверка всех возможных ветвей условных операторов.
* Покрытие путей: Тестирование всех возможных путей выполнения программы.
* Покрытие условий: Проверка всех логических условий в коде.

**Объектно-ориентированный подход:**

* Тестирование классов и методов: Проверка корректности работы отдельных классов и методов.
* Тестирование взаимодействий: Проверка взаимодействий между объектами.
* Инкапсуляция: Проверка корректного использования инкапсуляции и взаимодействия между компонентами через открытые интерфейсы.
* Статические методы тестирования
* Статическое тестирование проводится без выполнения кода, путем анализа документации, кода и других артефактов.

**Структурный подход**:

* Ревью кода: Проверка исходного кода на соответствие стандартам и выявление ошибок.
* Анализ кода (статический анализ): Использование инструментов статического анализа для выявления потенциальных дефектов в коде.
* Инспекции кода: Проверка кода другими разработчиками на предмет ошибок и соответствия стандартам кодирования.
* Проверка соответствия стандартам: Убедиться, что код соответствует установленным стандартам и требованиям.

**Объектно-ориентированный подход:**

* Ревью дизайна классов: Анализ дизайна классов и их взаимодействий для выявления потенциальных проблем.
* Инспекции кода: (Аналогично структурному подходу)
* Проверка UML-диаграмм: Анализ диаграмм классов, последовательностей и других UML-диаграмм для проверки соответствия требованиям.
* Анализ зависимостей: Проверка корректности зависимостей между классами и компонентами.

**Преимущества и недостатки**

**Динамическое тестирование:**

* Преимущества: Обнаружение дефектов во время выполнения программы, проверка реального поведения системы.
* Недостатки: Требует больше времени и ресурсов, зависит от качества тест-кейсов.

**Статическое тестирование:**

* Преимущества: Обнаружение дефектов на ранних стадиях разработки, экономия времени на исправление ошибок.

Недостатки: Не выявляет ошибки, возникающие во время выполнения программы.

\_\_\_\_\_

**I. Общие определения:**

Статическое тестирование: Проверка кода и сопутствующей документации без фактического выполнения программы. Цель - найти дефекты на ранних стадиях, предотвратить их появление на более поздних этапах разработки.

Динамическое тестирование: Проверка поведения программы во время её выполнения. Цель - найти дефекты, связанные с функциональностью, производительностью, безопасностью и другими аспектами работы.

**II. Структурный (Процедурный) Подход:**

В структурном программировании основное внимание уделяется функциям и процедурам, а данные обычно рассматриваются как отдельные сущности.

**Статическое тестирование (структурный подход):**

Инспекции кода: Проверка кода на соответствие стандартам кодирования, поиск логических ошибок, несоответствий в спецификациях, неиспользуемого кода, дублирования кода. Фокус на процедурах и функциях.

Обзоры кода: Более формальный процесс, чем инспекция, с участием нескольких экспертов.

Анализ потока данных: Отслеживание пути данных через программу для выявления неинициализированных переменных, неправильных операций, и т.д. Особенно важен для выявления проблем с глобальными переменными.

Анализ потока управления: Анализ порядка выполнения процедур и функций, выявление недостижимого кода, бесконечных циклов и т.д.

**Динамическое тестирование (структурный подход):**

Модульное тестирование: Проверка отдельных процедур или функций. Используются тестовые заглушки (stubs) и драйверы (drivers) для изоляции модулей.

Интеграционное тестирование: Проверка взаимодействия между различными процедурами и функциями. Варианты:

Сверху вниз (Top-down): Начиная с главной процедуры и постепенно интегрируя подчиненные.

Снизу вверх (Bottom-up): Начиная с самых низкоуровневых процедур и функций.

“Большой взрыв” (Big-bang): Интеграция всех модулей сразу.

Системное тестирование: Проверка всей системы в целом, на соответствие требованиям.

Тестирование производительности: Оценка времени выполнения процедур и функций, использования памяти и других ресурсов.

Нагрузочное тестирование: Проверка устойчивости системы при высокой нагрузке.

Тестирование стрессоустойчивости: Проверка поведения системы в экстремальных условиях (нехватка ресурсов, сбои оборудования).

**III. Объектно-ориентированный Подход:**

В объектно-ориентированном программировании основное внимание уделяется объектам, которые инкапсулируют данные и методы (функции), работающие с этими данными.

**Статическое тестирование (объектно-ориентированный подход):**

Инспекции кода: Проверка кода на соответствие принципам SOLID, правильность использования шаблонов проектирования, логические ошибки в методах классов, соответствие дизайна спецификациям. Фокус на классах, методах и их взаимодействии.

Обзоры кода: Аналогично структурному подходу, но с акцентом на объектно-ориентированные концепции.

Анализ диаграмм UML: Проверка диаграмм классов, последовательностей, состояний и других диаграмм UML на предмет согласованности и соответствия требованиям.

Анализ структуры классов: Проверка правильности использования наследования, полиморфизма, композиции и других объектно-ориентированных механизмов.

**Динамическое тестирование (объектно-ориентированный подход):**

Модульное тестирование (Классное тестирование): Проверка отдельных классов и их методов. Используются заглушки (mocks) для имитации зависимых объектов. Акцент на проверке инвариантов класса и корректной работы методов.

Интеграционное тестирование: Проверка взаимодействия между объектами и классами.

Тестирование кластеров: Группы связанных классов.

Тестирование подсистем: Больших частей системы, состоящих из нескольких кластеров.

Системное тестирование: Проверка всей системы в целом.

Тестирование состояний (State-based testing): Проверка правильности переходов между состояниями объектов.

Тестирование наследования: Проверка правильности работы методов, унаследованных от родительских классов. Особое внимание уделяется переопределенным методам.

Полиморфное тестирование: Проверка правильности работы полиморфных методов с различными типами объектов.

Тестирование производительности: Оценка производительности объектов и классов, времени ответа на сообщения.

Нагрузочное тестирование: Проверка устойчивости системы при высокой нагрузке.

Тестирование стрессоустойчивости: Проверка поведения системы в экстремальных условиях.

\_\_\_\_\_

Статические методы при структурном подходе включают использование инструментов и техник анализа кода без его запуска. Это может включать обзор кода, проверку форматирования и применение линтеров. Статические методы помогают выявить потенциальные проблемы на уровне кода до его выполнения.

Динамические методы при структурном подходе включают запуск программного кода и проверку его поведения во время выполнения. Это может включать тестирование на основе потока управления и потока данных, как описано выше.

При объектном подходе статические методы могут быть использованы для анализа структуры классов и их отношений, например, с помощью анализа UML-диаграмм. Динамические методы могут быть использованы для проверки поведения объектов во время выполнения программы.

\_\_\_\_\_

**Разработать модульные тесты для программного модуля Validation, содержащий метод проверки валидности email. Код для программного модуля находится «Resorce/ModuleC++/». Оформить тест-кейсы для модульного тестирования.**