**Модульное и интеграционное тестирование. Особенности модульного тестирования, подходы к тестированию на основе потока управления, потока данных. Организация модульного тестирования.**

**Модульное тестирование (Unit Testing):**

Фокусируется на проверке отдельных компонентов или модулей кода.

Изолирует модуль от других частей системы.

Цель - убедиться, что каждый модуль работает правильно сам по себе.

**Интеграционное тестирование (Integration Testing):**

Проверяет взаимодействие между модулями или компонентами.

Фокусируется на проверке потоков данных и управления между модулями.

Цель - убедиться, что модули работают вместе правильно.

**Особенности модульного тестирования:**

Изоляция: Модули тестируются изолированно, часто с использованием заглушек (stubs) и моков (mocks) для имитации зависимостей.

Автоматизация: Модульные тесты обычно автоматизированы и выполняются часто.

Фокус на деталях: Тестирование сосредоточено на проверке логики и алгоритмов внутри модуля.

**Подходы к тестированию на основе потока управления:**

Тестирование путей: Выбираются различные пути выполнения кода внутри модуля и создаются тесты для их проверки.

Тестирование условий: Проверяются различные условия (if/else, switch) и их влияние на поток управления.

Тестирование циклов: Проверяются циклы (for, while) с различными условиями и количеством итераций.

**Подходы к тестированию на основе потока данных:**

Тестирование границ входных данных: Проверяются граничные значения входных данных, которые могут повлиять на поток данных внутри модуля.

Тестирование недопустимых входных данных: Проверяется, как модуль обрабатывает недопустимые входные данные.

Тестирование промежуточных значений данных: Проверяются значения данных в различных точках внутри модуля.

**Организация модульного тестирования:**

Определение модулей для тестирования: Выберите модули, которые требуют тестирования.

Разработка тестовых сценариев: Создайте тестовые сценарии, которые покрывают различные аспекты работы модуля.

Создание заглушек и моков: Создайте заглушки и моки для имитации зависимостей модуля.

Написание тестов: Напишите автоматизированные тесты на основе тестовых сценариев.

Выполнение тестов: Запустите тесты и проверьте результаты.

Анализ результатов: Проанализируйте результаты тестирования и исправьте ошибки.

Повторное тестирование: Повторите тестирование после исправления ошибок.

**\_\_\_\_\_**

**Модульное тестирование**

Модульное тестирование (Unit Testing) предполагает проверку отдельных модулей или компонентов системы на их корректность. Модуль обычно представляет собой отдельную функцию, метод или класс.

**Особенности модульного тестирования:**

Изолированность тестирования: Каждый модуль тестируется отдельно от других, что позволяет выявлять ошибки именно в этом модуле.

Автоматизация: Часто используется автоматизированное тестирование с помощью фреймворков (JUnit, NUnit, pytest и др.).

Раннее выявление дефектов: Позволяет обнаруживать ошибки на ранних этапах разработки.

Легкость в исполнении: Быстрое выполнение тестов из-за их ограниченного объема.

**Подходы к тестированию на основе потока управления и потока данных:**

**Поток управления** (Control Flow Testing): Тестирование направлено на покрытие всех возможных путей выполнения программы, включая условия и циклы. Основные метрики:

Покрытие операторов: Проверка выполнения всех операторов в коде.

Покрытие ветвей: Проверка выполнения всех ветвей условных операторов.

Покрытие путей: Тестирование всех возможных путей выполнения программы.

**Поток данных** (Data Flow Testing): Анализ использования данных в программе, включая их инициализацию, модификацию и использование. Основные метрики:

Def-Use Chain: Проверка цепочек определения и использования переменных.

Du-Path: Тестирование всех возможных путей между определением и использованием данных.

**Организация модульного тестирования**

Разработка тест-кейсов: Создание сценариев тестирования для каждого модуля, включающих входные данные, ожидаемые результаты и шаги выполнения.

Написание тестов: Реализация тестов с использованием выбранного фреймворка.

Подготовка тестовой среды: Настройка окружения для выполнения тестов, включающего необходимые библиотеки и зависимости.

Исполнение тестов: Запуск тестов и фиксация результатов.

Анализ результатов: Оценка результатов тестирования и документирование выявленных дефектов.

Регресс-тестирование: Повторное тестирование после исправления дефектов для проверки, что изменения не вызвали новых проблем.

**Интеграционное тестирование**

Интеграционное тестирование (Integration Testing) направлено на проверку взаимодействия между различными модулями системы.

Основные особенности:

Тестирование взаимодействия: Проверка корректности взаимодействия между модулями.

Использование тестовых сред: Создание тестовых сред, имитирующих реальную работу системы.

Этапы тестирования:

* Тестирование больших интеграционных блоков: Проверка взаимодействия между группами модулей.
* Тестирование системы целиком: Проверка всей системы в целом.

\_\_\_\_\_

Тестирование на основе потока управления - это метод тестирования, который основывается на проверке различных вариантов исполнения программы в зависимости от переходов в управляющих конструкциях (например, условных операторах или циклах).

Тестирование на основе потока данных - это метод тестирования, который фокусируется на проверке потока данных в программе

\_\_\_\_\_

**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.**