Выполнение интеграционного тестирования программы.

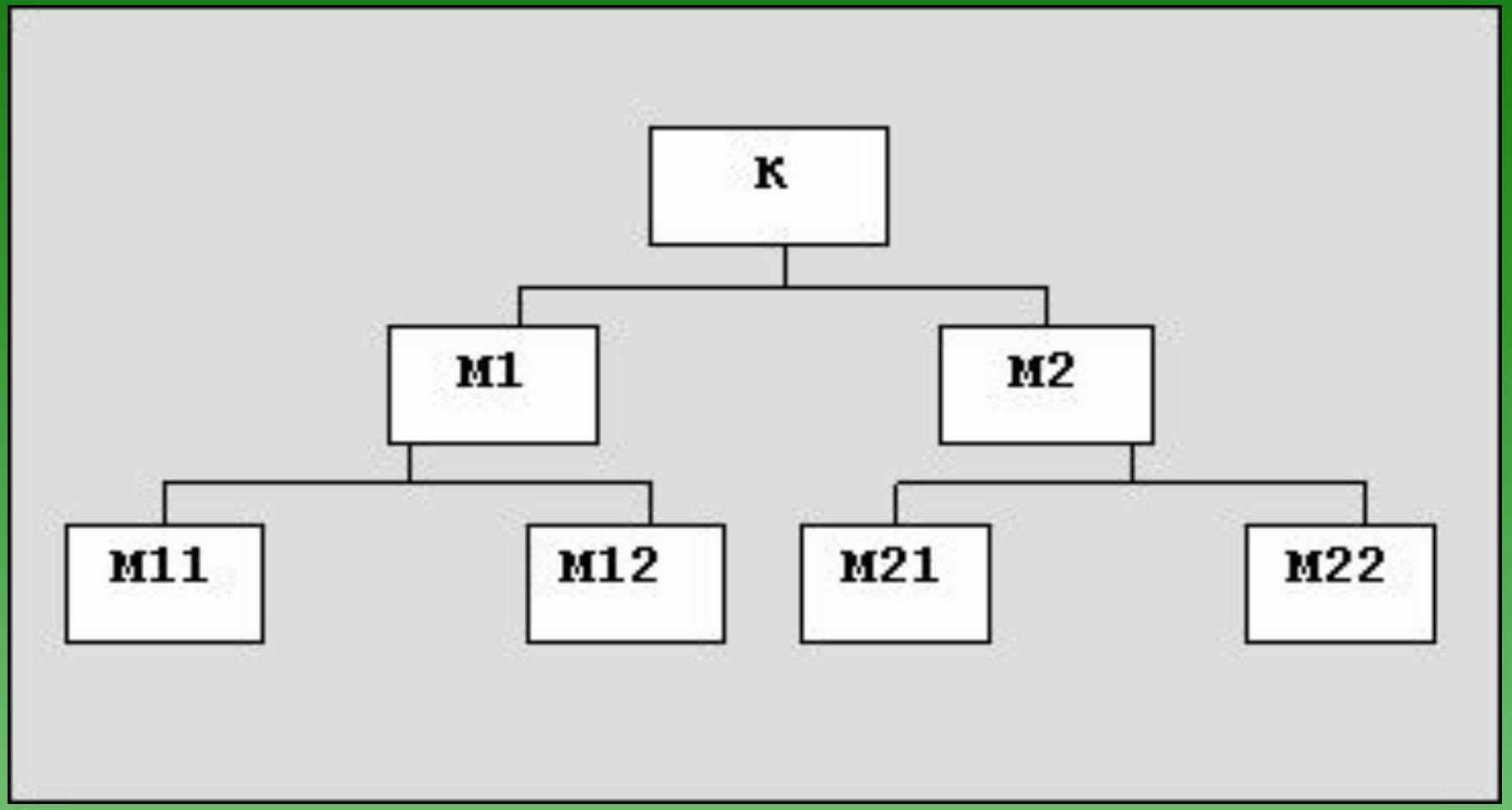
Интеграционное тестирование

- ***Интеграционное тестирование*** - это тестирование части системы, состоящей из двух и более модулей. Основная задача *интеграционного тестирования* - поиск дефектов, связанных с ошибками в реализации и интерпретации интерфейсного взаимодействия между модулями.

- С технологической точки зрения *интеграционное тестирование* является количественным развитием *модульного*, поскольку так же, как и *модульное тестирование*, оперирует интерфейсами модулей и подсистем и требует создания тестового окружения, включая *заглушки*(**Stub**) на месте

- Основная разница между *модульным* и *интеграционным тестированием* состоит в целях, то есть в типах обнаруживаемых дефектов, которые, в свою *очередь*, определяют стратегию выбора входных данных и методов анализа. В частности, на уровне *интеграционного тестирования* часто применяются методы, связанные с покрытием интерфейсов, например, вызовов функций или методов, или *анализ* использования интерфейсных объектов, таких как глобальные ресурсы, средства коммуникаций, предоставляемых операционной системой

Пример структуры комплекса программ



- На рисунке приведена структура комплекса программ К, состоящего из оттестированных на этапе *модульного тестирования* модулей М1, М2, М11, М12, М21, М22. Задача, решаемая методом *интеграционного тестирования,* - тестирование межмодульных связей, реализующихся при исполнении программного обеспечения комплекса К.
- *Интеграционное тестирование* использует модель "белого ящика" на модульном уровне. Поскольку тестировщику текст программы известен с детальностью до вызова всех модулей, входящих в тестируемый комплекс, применение структурных критериев на данном этапе возможно и оправдано.

- *Интеграционное тестирование* применяется на этапе сборки модульно оттестированных модулей в единый комплекс. Известны два метода *сборки модулей*:
- **Монолитный**, характеризующийся одновременным объединением всех модулей в тестируемый комплекс
- **Инкрементальный**, характеризующийся пошаговым (помодульным) наращиванием комплекса программ с **пошаговым тестированием** собираемого комплекса. В инкрементальном методе выделяют две стратегии добавления модулей:
  - "Сверху вниз" и соответствующее ему *нисходящее тестирование*.
  - "Снизу вверх" и соответственно *восходящее тестирование*.

- **Особенности монолитного тестирования** заключаются в следующем: для замены неразработанных к моменту тестирования модулей, кроме самого верхнего, как на рисунке, необходимо дополнительно разрабатывать **драйверы** (**test driver**) и/или **заглушки** (**stub**) , замещающие отсутствующие на момент сеанса тестирования модули нижних уровней.

- Сравнение *монолитного* и инкрементального подхода дает следующее:
- *Монолитное тестирование* требует больших трудозатрат, связанных с дополнительной разработкой драйверов и заглушек и со сложностью идентификации ошибок, проявляющихся в пространстве собранного кода.
- Пошаговое тестирование связано с меньшей трудоемкостью идентификации ошибок за счет постепенного наращивания объема тестируемого кода и соответственно локализации добавленной области тестируемого кода.
- *Монолитное тестирование* предоставляет большие возможности распараллеливания работ особенно на начальной *фазе тестирования*.

- Особенности *нисходящего тестирования* заключаются в следующем: организация среды для исполняемой очередности вызовов оттестированными модулями тестируемых модулей, постоянная разработка и использование заглушек, организация приоритетного тестирования модулей, содержащих *операции* обмена с окружением, или модулей, критичных для тестируемого алгоритма.

- НЕДОСТАТКИ *НИСХОДЯЩЕГО ТЕСТИРОВАНИЯ*:
- Проблема разработки достаточно "интеллектуальных" заглушек, т.е. заглушек, пригодных к использованию при моделировании различных режимов работы комплекса, необходимых для тестирования
- Сложность организации и разработки среды для реализации исполнения модулей в нужной последовательности
- Параллельная разработка модулей верхних и нижних уровней приводит к не всегда эффективной реализации модулей из-за подстройки (специализации) еще не тестированных модулей нижних уровней к уже оттестированным модулям верхних уровней

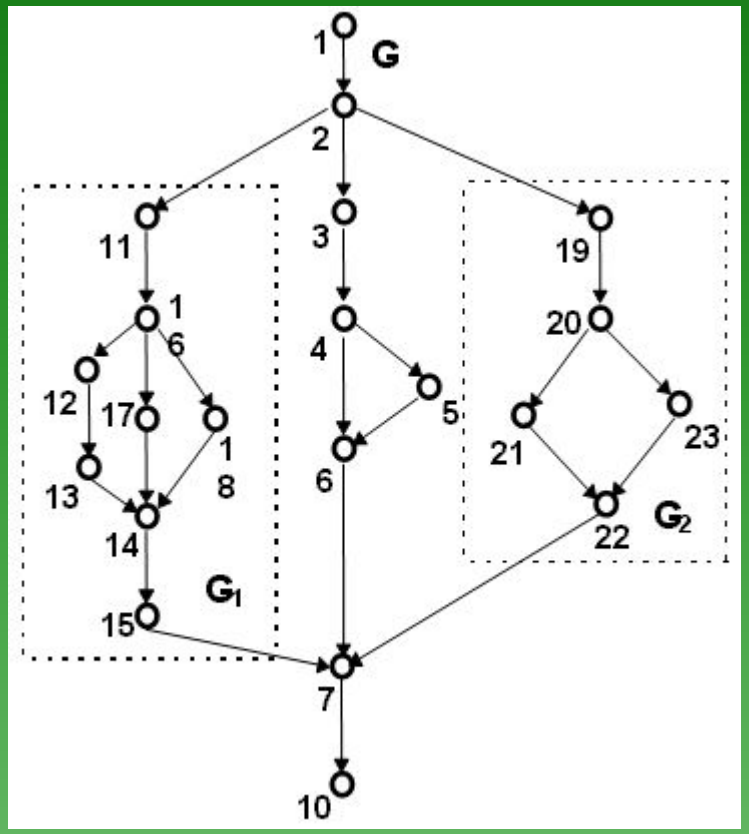
- **Особенности восходящего тестирования** в организации порядка *сборки* и перехода к тестированию модулей, соответствующему порядку их реализации.
- Недостатки *восходящего тестирования*:
- Запаздывание проверки концептуальных особенностей тестируемого комплекса
- Необходимость в разработке и использовании драйверов

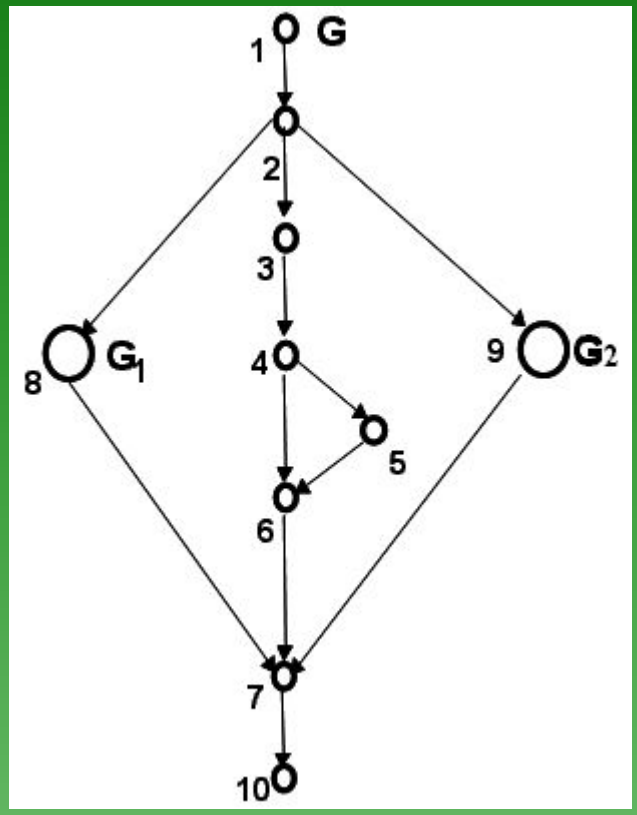
Особенности интеграционного тестирования для процедурного программирования

- Для *интеграционного тестирования* наиболее существенным является рассмотрение *модели программы*, построенной с использованием диаграмм *потоков управления*. Контролируются также связи через данные, подготавливаемые и используемые другими группами программ при взаимодействии с тестируемой группой. Каждая переменная межмодульного интерфейса проверяется на тождественность описаний во взаимодействующих модулях, а также на соответствие исходным программным спецификациям. Состав и структура информационных связей реализованной группы модулей проверяются на соответствие спецификации требований этой группы. Все реализованные связи должны быть установлены, упорядочены и обобщены.

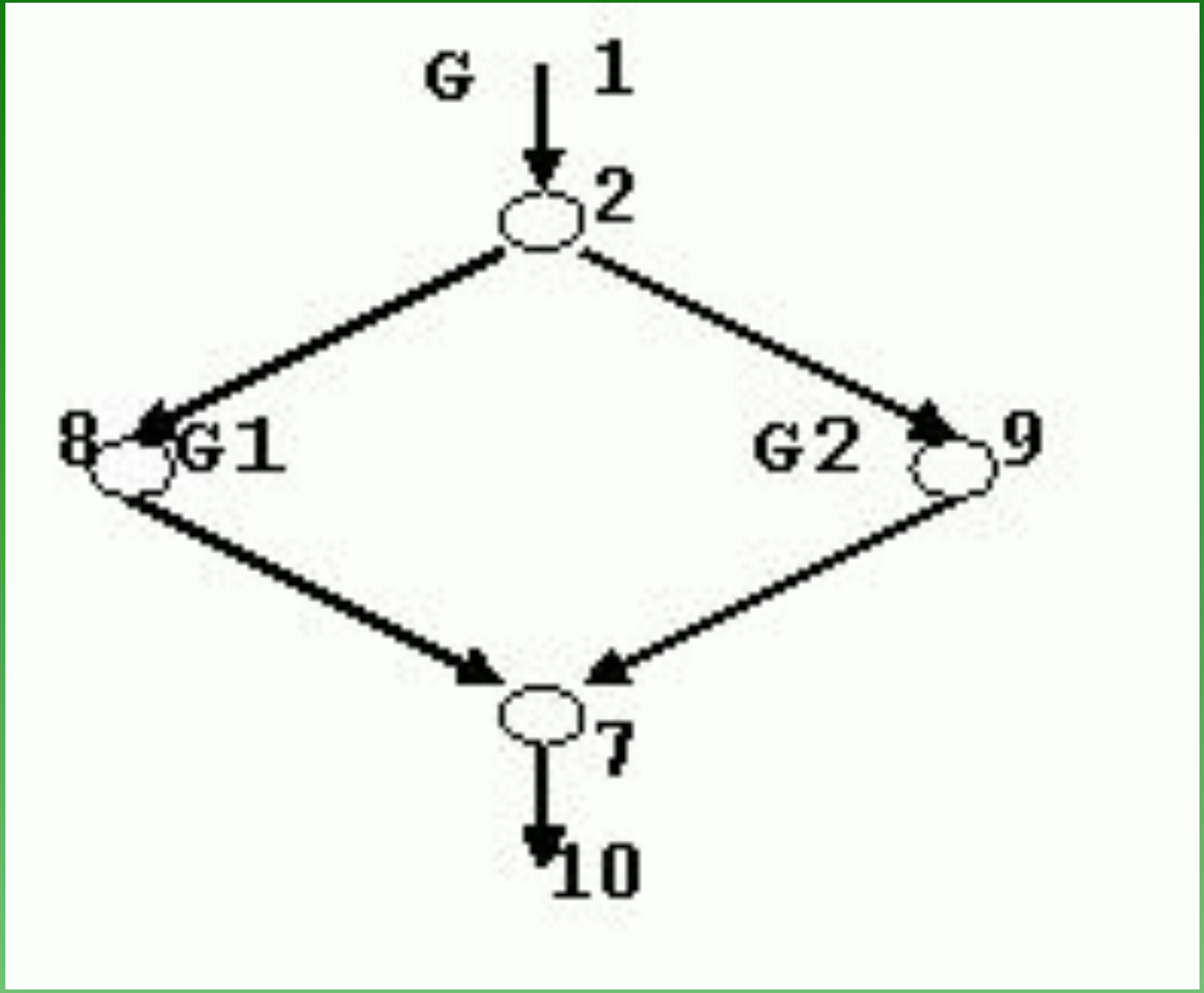
- При *сборке модулей* в единый программный комплекс появляется два варианта построения *графовой модели* проекта:
- Плоская или иерархическая модель проекта
- *Граф вызовов*.

Плоская или иерархическая модель проекта





Граф вызовов



Интеграционное тестирование включает в себя следующие этапы

- составление тест-плана,
- создание тест-кейсов и юз-кейсов,
- выполнение тестов после интеграции модулей,
- выявление ошибок и повторное тестирование после их исправления.
- Мы повторяем цикл тестирования до тех пор, пока все баги не будут исправлены.