# Демонстрация конкретных примеров понятия отладки и тестирования

**Тестирование** - это процесс выполнения программы, целью которого является выявление ошибок. Никакое тестирование не может доказать отсутствие ошибок в сложном ПО, поскольку выполнение полного тестирования становится невозможным и имеется вероятность, что остались невыявленные ошибки. Соблюдение основных правил тестирования и научно обоснованный подбор тестов может уменьшить их количество. Процесс разработки согласно современной модели жизненного цикла ПО предполагает три стадии тестирования: *автономное* тестирование компонентов ПО; *комплексное* тестирование разрабатываемого ПО; *системное* или *оценочное* тестирование на соответствие основным критериям качества. Для повышения качества тестирования рекомендуется соблюдать следующие основные принципы:

1. предполагаемые результаты должны быть известны до тестирования;
2. следует избегать тестирования программы автором;
3. необходимо досконально изучать результаты каждого теста;
4. необходимо проверять действия программы на неверных данных;
5. необходимо проверять программу на неожиданные побочные эффекты на неверных данных.

Вероятность наличия необнаруженных ошибок в части программы пропорциональна количеству ошибок уже найденных в этой части. Удачным считают тест, который обнаруживает хотя бы одну ошибку. Формирование набора тестов имеет большое значение, поскольку тестирование является одним из наиболее трудоемких этапов создания ПО. Доля стоимости тестирования в общей стоимости разработки возрастает при увеличении сложности ПО и повышении требований к их качеству.

Существуют два принципиально различных подхода к формированию тестовых наборов: *структурный* и *функциональный*. **Структурный подход** базируется на том, что известна структура тестируемого ПО, в том числе его алгоритмы («*стеклянный ящик*»). Тесты строятся для проверки правильности реализации заданной логики в коде программы. **Функциональный подход** основывается на том, что структура ПО не известна («*черный ящик*»). В этом случае тесты строят, опираясь на функциональные спецификации. Этот подход называют также *подходом, управляемым данными,* так как при его использовании тесты строят на базе различных способов декомпозиции множества данных. Наборы тестов, полученные в соответствии с методами этих подходов, объединяют, обеспечивая всестороннее тестирование ПО.

**Ручной контроль** используют на ранних этапах разработки. Все проектные решения анализируются с точки зрения их правильности и целесообразности как можно раньше, пока их можно легко пересмотреть. Различают *статический* и *динамический* подходы к ручному контролю. При статическом подходе анализируют структуру, управляющие и информационные связи программы, ее входные и выходные данные. При *динамическом* - выполняют *ручное тестирование* (вручную моделируют процесс выполнения программы на заданных исходных данных). Исходными данными для таких проверок являются: техническое задание, спецификации, структурная и функциональная схемы программного продукта, схемы отдельных компонентов, а для более поздних этапов - алгоритмы и тексты программ, а также тестовые наборы. Доказано, что ручной контроль способствует существенному увеличению производительности и повышению надежности программ и с его помощью можно находить от 30 до 70 % ошибок логического проектирования и кодирования. Основными методами ручного контроля являются: *инспекции исходного текста*, *сквозные просмотры*, *проверка за столом*, *оценки программ*.

В основе **структурного тестирования** лежит концепция максимально полного тестирования всех *маршрутов*, предусмотренных алгоритмом (последовательности операторов программы, выполняемых при конкретном варианте исходных данных). Недостатки: построенные тестовые наборы не обнаруживают пропущенных маршрутов и ошибок, зависящих от заложенных данных; не дают гарантии, что программа правильна.

Другим способом проверки программ является **функциональное тестирование**: программа рассматривается как «*черный ящик*», целью тестирования является выяснение обстоятельств, когда поведение программы не соответствует спецификации. Для обнаружения всех ошибок необходимо выполнить *исчерпывающее* тестирование (при всех возможных наборах данных), что для большинства случаев невозможно. Поэтому обычно выполняют «*разумное*» или «*приемлемое*» тестирование, ограничивающееся прогонами программы на небольшом подмножестве всех возможных входных данных. При функциональном тестировании различают следующие методы формирования тестовых наборов: *эквивалентное разбиение*; *анализ граничных значений*; *анализ причинно-следственных связей*; *предположение об ошибке*.

При **комплексном тестировании** используют тесты, построенные по методам эквивалентных классов, граничных условий и предположении об ошибках, поскольку структурное тестирование для него не применимо. Одним из самых сложных является вопрос о завершении тестирования, так как невозможно гарантировать, что в программе не осталось ошибок. Часто тестирование завершают потому, что закончилось время, отведенное на его выполнение. Его сворачивают, обходясь **минимальным тестированием** [15], которое предполагает: тестирование граничных значений, тщательную проверку руководства, тестирование минимальных конфигураций технических средств, возможности редактирования команд и повторения их в любой последовательности, устойчивости к ошибкам пользователя.

После завершения комплексного тестирования приступают к **оценочному тестированию**, целью которого является поиск несоответствий техническому заданию. Оценочное тестирование включает тестирование: удобства использования, на предельных объемах, на предельных нагрузках, удобства эксплуатации, защиты, производительности, требований к памяти, конфигурации оборудования, совместимости, удобства установки, удобства обслуживания, надежности, восстановления, документации, процедуры.

**Отладка** - это процесс *локализации* (определения оператора программы, выполнение которого вызвало нарушение вычислительного процесса) и исправления ошибок, обнаруженных при тестировании ПО. Для исправления ошибки необходимо определить ее причину. Отладка требует от программиста глубоких знаний специфики управления используемыми техническими средствами, операционной системы, среды и языка программирования, реализуемых процессов, природы и специфики ошибок, методик отладки и соответствующих программных средств; психологически дискомфортна (нужно искать собственные ошибки в условиях ограниченного времени); оставляет возможность взаимовлияния ошибок в разных частях программы. Четко сформулированные методики отладки отсутствуют. Различают:

1. *синтаксические ошибки* – сопровождаются комментарием с указанием их местоположения, фиксируются компилятором (транслятором) при выполнении синтаксического и частично семантического анализа;
2. *ошибки компоновки* - обнаруживаются компоновщиком (редактором связей) при объединении модулей программы;
3. *ошибки выполнения* - обнаруживаются аппаратными средствами, операционной системой или пользователем при выполнении программы, проявляются разными способами и в свою очередь делятся на группы:

1. ошибки определения исходных данных (ошибки передачи, ошибки преобразования, ошибки перезаписи и ошибки данных);
2. логические ошибки проектирования (неприменимый метод, неверный алгоритм, неверная структура данных, другие) и кодирования (ошибки некорректного использования переменных, вычислений, межмодульного интерфейса, реализации алгоритма, другие);
3. ошибки накопления погрешностей результатов вычислений (игнорирование ограничений разрядной сетки и способов уменьшения погрешности).

Отладка программы в любом случае предполагает обдумывание и логическое осмысление всей имеющейся информации об ошибке. Большинство ошибок можно обнаружить по косвенным признакам посредством тщательного анализа текстов программ и результатов тестирования без получения дополнительной информации с помощью следующих методов:

1. *ручного тестирования* (при обнаружении ошибки нужно выполнить тестируемую программу вручную, используя тестовый набор, при работе с которым была обнаружена ошибка);
2. *индукции* (основан на тщательном анализе симптомов ошибки, которые могут проявляться как неверные результаты вычислений или как сообщение об ошибке);
3. *дедукции* (вначале формируют множество причин, которые могли бы вызвать данное проявление ошибки, а затем анализируя причины, исключают те, которые противоречат имеющимся данным);
4. *обратного прослеживания* (для точки вывода неверного результата строится гипотеза о значениях основных переменных, которые могли бы привести к получению данного результата, а

затем, исходя из этой гипотезы, делают предположения о значениях переменных в предыдущей точке).

Для получения дополнительной информации об ошибке выполняют добавочные тесты и используют специальные методы и средства: *отладочный вывод*; *интегрированные средства отладки*; *независимые отладчики*.

Общая методика отладки программных продуктов, написанных для выполнения в операционных системах MS DOS и Win32:

*1 этап* - изучение проявления ошибки;

*2 этап* – определение локализации ошибки;

*3 этап* - определение причины ошибки;

*4 этап* — исправление ошибки;

*5 этап* - повторное тестирование.

Процесс отладки можно существенно упростить, если следовать основным рекомендациям структурного подхода к программированию:

1. программу наращивать «сверху-вниз», от интерфейса к обрабатывающим подпрограммам, тестируя ее по ходу добавления подпрограмм;
2. выводить пользователю вводимые им данные для контроля и проверять их на допустимость сразу после ввода;
3. предусматривать вывод основных данных во всех узловых точках алгоритма (ветвлениях, вызовах подпрограмм).