



БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ
И РАДИОЭЛЕКТРОНИКИ

ТЕСТИРОВАНИЕ ВЕБ-ОРИЕНТИРОВАННЫХ ПРИЛОЖЕНИЙ ПЛАНИРОВАНИЕ ИСТЫТАНИЙ. РАЗРАБОТКА ТЕСТОВ

ШУЛЬДОВА СВЕТЛАНА ГЕОРГИЕВНА

ЦЕЛЬ И СОДЕРЖАНИЕ ЛЕКЦИИ

- **Цель** лекции – рассмотреть порядок разработки и содержание тестовой документации этапов планирования и разработки тестов и изучить техники тестирования.
- **План лекции**
 1. Планирование испытаний. Тестовый план
 2. Разработка тестов. Техники тестирования
 3. Критерии тестирования

ОСНОВНАЯ ЛИТЕРАТУРА



1. **Майерс, Г.** Искусство тестирования программ / Г. Майерс, Т. Баджетт, К. Сандлер . — М.: Дialeктика, 2016. — С. 54-95.



2. **Куликов, С.С.** Тестирование программного обеспечения. Базовый курс : Практик. Пособие /Куликов С.С. — Минск: Четыре четверти, 2017. — С. 67-69, 223-233.



3. **Бейзер, Б.** Тестирование черного ящика. Технологии функционального тестирования программного обеспечения и систем / Б. Бейзер. — СПб.: Питер, 2004. — С. 57-90.

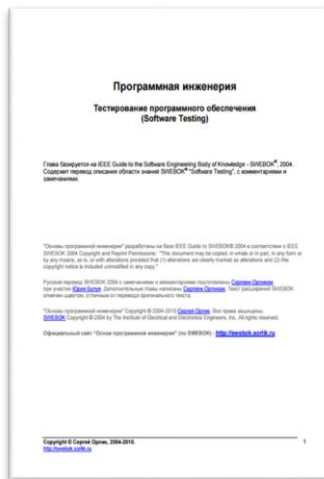


4. **Калбертсон, Р.** Быстрое тестирование / Р. Калбертсон, К. Браун, Гэри Кобб. — СПб.: Вильямс, 2002. . — С. 56-118

СТАНДАРТЫ



5. Стандартный глоссарий терминов, используемых в тестировании программного обеспечения. Версия 2.3 (от 9 июля 2014 года) / 'Glossary Working Party' International Software Testing Qualifications Board.



6. Программная инженерия. Тестирование программного обеспечения (Software Testing) / IEEE Guide to the Software Engineering Body of Knowledge - SWEBOK®, 2004.

ПЛАНИРОВАНИЕ ИСПЫТАНИЙ

Планы – бесполезны,
планирование – бесценно .

Дуайт Эзенхауэр



ПЛАНИРОВАНИЕ ИСПЫТАНИЙ

- **Планирование** — непрерывный процесс принятия управленческих решений и методической организации усилий по их реализации с целью обеспечения качества некоторого процесса на протяжении длительного периода времени.





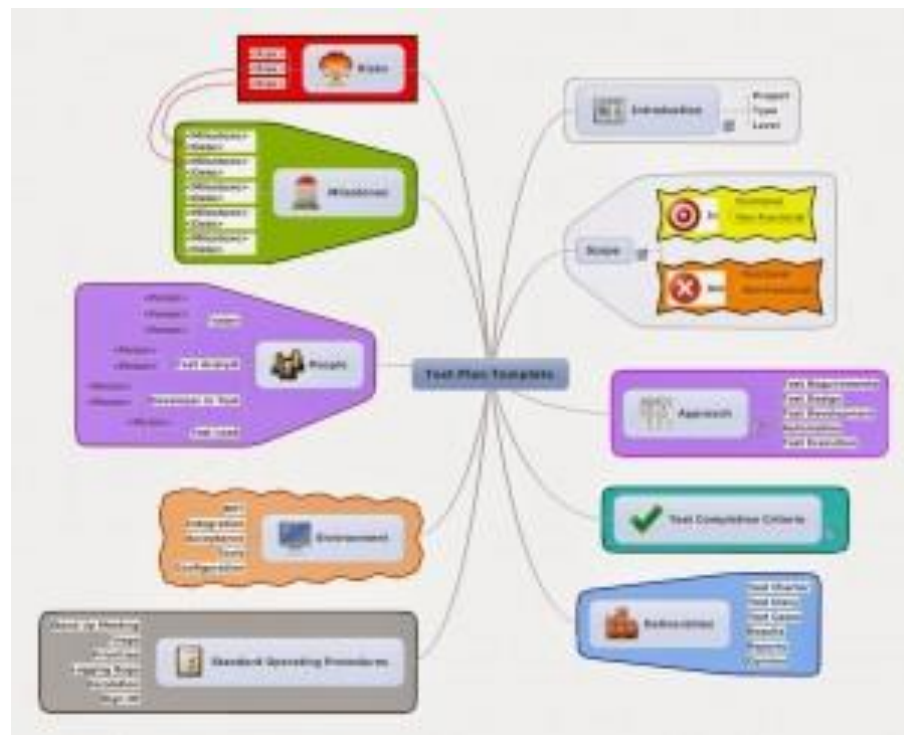
ОПРЕДЕЛЕНИЕ

План тестирования (test plan) – документ, описывающий цели, подходы, ресурсы и график запланированных тестовых мероприятий.

1. Цель
2. Список тестируемых/нетестируемых функций
3. Стратегия тестирования (может быть отдельный документ)
4. Ресурсы
5. Календарный план работ по тестированию
6. **Критерии**
7. Риски
8. Документация и **метрики** [IEEE 829]

ПОЯСНЕНИЕ

1. Документ, содержащий сведения «кто, что, когда и как будет тестировать».
2. Документ, содержащий все необходимые для проверки приложения тесты.



ОПРЕДЕЛЕНИЯ

- **СТРАТЕГИЯ** (др.-греч. *στρατηγία*, «искусство полководца») — план или модель действий, предназначенные для достижения целей.
- **МЕТОД** (от греч. *methodos* - путь исследования) — способ достижения цели либо совокупность приемов или операций для достижения цели или решения конкретной задачи.

Энциклопедический словарь

ОПРЕДЕЛЕНИЯ

Стратегия тестирования – совокупность систематических методов отбора, создания и реализации тестов.

- Методы тестирования
- Уровни тестирования
- Виды тестирования
- Инструментальные средства

МЕТОД «ЧЕРНОГО ЯЩИКА»

- Тестирование «**черного ящика**» (*black-box testing*) — метод тестирования, при которой программа рассматривается как объект, внутренняя структура которого неизвестна.
- Тесты основаны на требованиях*, четко зафиксированных в спецификациях.



**BLACK
BOX**

*— и здравого смысла для случаев, когда поведение программы в некоторой ситуации явно не регламентировано

МЕТОД «БЕЛОГО ЯЩИКА»

- Тестирование «**белого ящика**» (*white-box testing*) — метод тестирования, который позволяет исследовать внутреннюю структуру программы ⇒ **структурное тестирование**.
- Тесты основаны на знании кода приложения и его внутренних механизмов.



WHITE
BOX

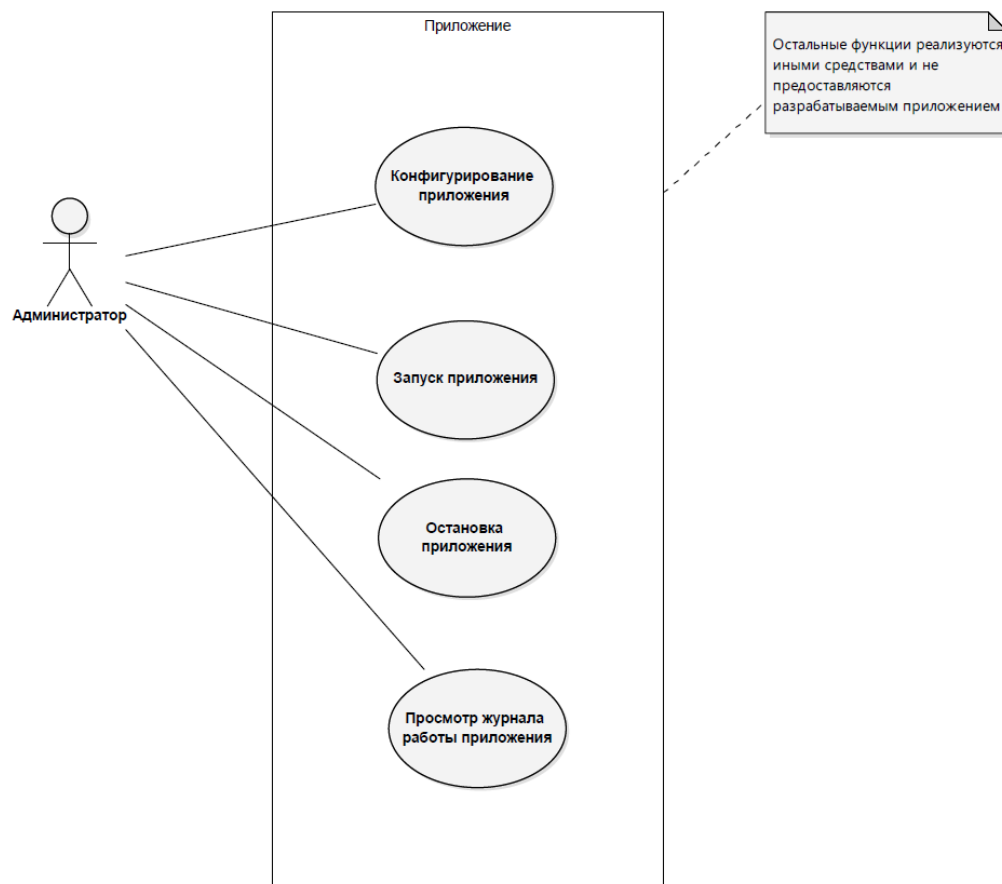
МЕТОД «СЕРОГО ЯЩИКА»

- Тестирование «серого ящика» (*gray-box testing*) — метод тестирования, сочетающий методы «белого ящика» и «чёрного ящика».
- Тесты основаны на спецификациях с учетом знания части исходного кода и архитектуры.



ПРИМЕР [2]

ЦЕЛЬ: Корректное автоматическое преобразование содержимого документов к единой кодировке с производительностью, значительно превышающей производительность человека при выполнении аналогичной задачи.



ПРИМЕР ТЕСТОВОГО ПЛАНА [2]

- Тестовая стратегия и подходы

- Общий подход.

Специфика работы приложения состоит в однократном конфигурировании опытным специалистом и дальнейшем использовании конечными пользователями, для которых доступна только одна операция — размещение файла в каталоге-приёмнике. Потому вопросы удобства использования, безопасности и т. п. не исследуются в процессе тестирования.

- Уровни функционального тестирования:

- Дымовой тест: автоматизированный с использованием командных файлов ОС Windows и Linux.
- Тест критического пути: выполняется вручную.
- Расширенный тест не производится, т. к. для данного приложения вероятность обнаружения дефектов на этом уровне пренебрежимо мала.

ПРИМЕР [2]

Требования

- ПТ-1: Запуск и остановка приложения.
 - ПТ-1.1: Запуск приложения производится из консоли командой «PHP converter.php параметры».
 - ПТ-1.2: Остановка приложения производится выполнением команды Ctrl+C.
- ПТ-2: Конфигурирование приложения.
- ПТ-2.1: Конфигурирование приложения сводится к указанию путей в файловой системе.
- ПТ-2.2: Целевой кодировкой является UTF8.

Области, подвергаемые тестированию

- ПТ-1.*: дымовой тест.
- ПТ-2.*: дымовой тест, тест критического пути.

■ Критерии

- Приёмочные критерии: успешное прохождение 100 % тест-кейсов уровня дымового тестирования и 90 % тест-кейсов уровня критического пути при условии устранения 100 % дефектов критической и высокой важности. Итоговое покрытие требований тест-кейсами требований тест-кейсами») должно составлять не менее 80 %.

- Критерии начала тестирования: выход билда.

- Критерии приостановки тестирования: переход к тесту критического пути допустим только при успешном прохождении 100 % тест-кейсов дымового теста;

тестирование может быть приостановлено в случае, если при выполнении не менее 25 % запланированных тест-кейсов более 50 % из них завершились обнаружением дефекта .

- Критерии возобновления тестирования: исправление более 50 % обнаруженных на предыдущей итерации дефектов .

- Критерии завершения тестирования: выполнение более 80 % запланированных на итерацию тест-кейсов.

ПРИМЕР [2]

- Тестовая стратегия и подходы

- Общий подход.

Специфика работы приложения состоит в однократном конфигурировании опытным специалистом и дальнейшем использовании конечными пользователями, для которых доступна только одна операция — размещение файла в каталоге-приёмнике. Потому вопросы удобства использования, безопасности и т. п. не исследуются в процессе тестирования.

- Уровни функционального тестирования:

- Дымовой тест: автоматизированный с использованием командных файлов ОС Windows и Linux.
- Тест критического пути: выполняется вручную.
- Расширенный тест не производится, т. к. для данного приложения вероятность обнаружения дефектов на этом уровне пренебрежимо мала.

КРИТЕРИИ ТЕСТИРОВАНИЯ

Критерии качества

Показатели качества, которым разрабатываемый продукт должен соответствовать, чтобы быть готовым к эксплуатации.

Критерии начала тестирования

Перечень условий, при выполнении которых команда приступает к тестированию.

Критерии приостановки тестирования

Перечень условий, при выполнении которых тестирование приостанавливается.

Критерии возобновления тестирования

Перечень условий, при выполнении которых тестирование возобновляется.

Критерии окончания тестирования

Перечень условий, при выполнении которых тестирование завершается.

МЕТРИКИ

- **Метрика** — числовая характеристика показателя качества. Может включать описание способов оценки и анализа результата.
- **Покрытие** — процентное выражение степени, в которой исследуемый элемент затронут соответствующим набором тест-кейсов.
- **Метрики покрытия**
 - Тестовое покрытие, основанное на покрытии требований
 - Тестовое покрытие, основанное на покрытии кода приложения

ФУНКЦИОНАЛЬНЫЕ КРИТЕРИИ: ПРИМЕР

- **Покрывтие требований** (*Requirements Coverage*) – оценка покрытия тестами функциональных и нефункциональных требований к продукту путем построения матриц трассировки (покрытия):

$$T_{cov} = \frac{L_{cov}}{L_{total}} \cdot 100\%$$

где:

T_{cov} – тестовое покрытие;

L_{cov} – количество требований, проверяемых тест-кейсами;

L_{total} – общее количество требований.

ПРИМЕР

Задача о треугольнике:

Программа производит чтение трёх целых чисел, которые интерпретируются как длины сторон треугольника. Далее программа печатает сообщение о том, является ли треугольник разносторонним, равнобедренным или равносторонним.

Г. Майерс, 1979

ПРИМЕР

1. Корректный разносторонний треугольник
2. Корректный равносторонний треугольник
3. Три корректных равнобедренных треугольника ($a=b$, $b=c$, $a=c$)
4. Длина одной из сторон равна нулю
5. Длина одной из сторон принимает отрицательное значение
6. Не выполняется правило треугольника (три варианта $a+b=c$, $a+c=b$, $b+c=a$; три варианта $a+b<c$, $a+c<b$, $b+c<a$)
7. Длина одной из сторон не является числом (буква, специальный символ)
8. Длина одной из сторон – вещественное число
9. Неверное количество аргументов



РАЗРАБОТКА ТЕСТОВ. ТЕХНИКИ ТЕСТИРОВАНИЯ

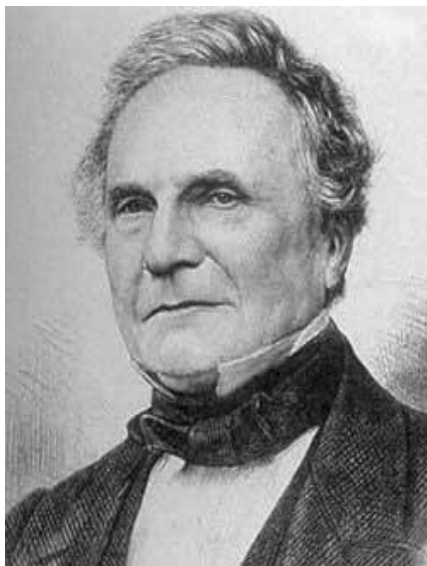


Меня два раза спрашивали [члены Парламента]:
«Скажите на милость, мистер Бэббидж, что случится, если вы введете в машину неверные цифры? Сможем ли мы получить правильный ответ?» Я не могу себе даже представить, какая путаница в голове может привести к подобному вопросу.

Чарльз Беббидж

ЧАРЛЬЗ БЭББИДЖ

26 декабря 1791- 18 октября 1871



Английский математик и инженер, автор трудов по теории функций, механизации счета в экономике; иностранный член-корреспондент Петербургской АН (1832).
В 1833 разработал проект аналитической машины - прообраза ЭВМ.

ОПРЕДЕЛЕНИЯ

- **Тест** — набор из одного или нескольких тест-кейсов.
- **Тест-кейс** (тестовый случай) — набор входных данных, условий выполнения и ожидаемых результатов, разработанный с целью проверки того или иного свойства или поведения программного средства [IEEE Std 829-1983].

Высокоуровневый

- Без конкретных входных данных и ожидаемых результатов

Низкоуровневый

- Содержит конкретные входные данные и ожидаемые результаты

Тест-сценарий — документ, описывающий последовательность действий по выполнению теста.

АТТРИБУТЫ ТЕСТ-КЕЙСА

- Идентификатор теста (**id**).
- Связанное с тестом требование (**related requirement**).
- Краткое заглавие теста (**title**).
- Модуль и подмодуль приложения, к которым относится тест (**module, submodule**).
- Приоритет теста (**priority: smoke, critical, extended;**).
- Исходные данные, необходимые для теста (**initial data**).
- Шаги для выполнения теста (**steps**).
- Ожидаемые результаты (**expected results**).
- Отвести поле для пометки, прошёл тест или нет (**passed or failed**).
- Указать автора теста (**author**), время последнего выполнения теста (**last time run**), последний полученный актуальный результат (**actual result**).

Приоритет	Связанное с тестом требование			Заглавие (суть) теста		Ожидаемый результат по каждому шагу
UG_U 1.12	A	R97	Галерея	Загрузка файла	<p>Галерея, загрузка файла, имя со спецсимволами</p> <p>Приготовление: создать непустой файл с именем #\$\$%^&.jpg</p> <p>1. Нажать кнопку «Загрузить картинку»</p> <p>2. Нажать кнопку «Загрузить файл»</p> <p>3. Выбрать из списка подготовленный файл</p> <p>4. Нажать кнопку «ОК»</p> <p>5. Нажать кнопку «Добавить в галерею»</p>	<p>1. Появляется окно загрузки картинки</p> <p>2. Появляется диалоговое окно браузера выбора файла для загрузки</p> <p>3. Имя выбранного файла появляется в поле «Файл»</p> <p>4. Диалоговое окно файла закрывается, в поле «Файл» появляется полное имя файла</p> <p>5. Выбранный файл появляется в списке файлов галереи</p>

Идентификатор

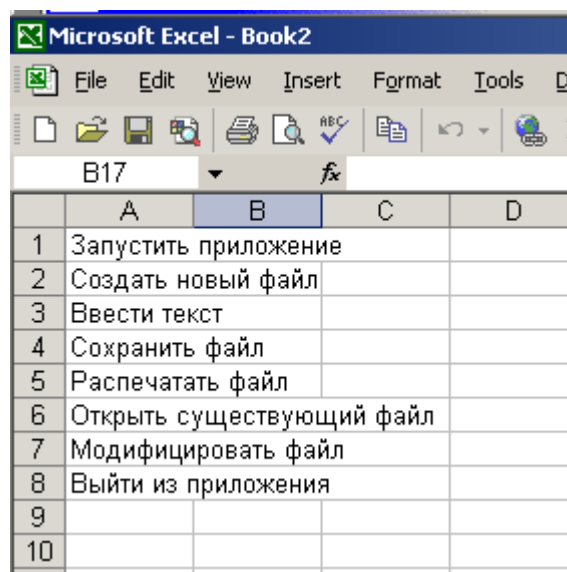
Модуль и подмодуль

Исходные данные, необходимые для выполнения теста

Шаги

ПРИМЕР РАЗРАБОТКИ ТЕСТОВ

Smoke test Notepad



The screenshot shows a Microsoft Excel spreadsheet titled "Microsoft Excel - Book2". The spreadsheet contains a list of test steps for a smoke test of Notepad. The steps are listed in column A, with corresponding test data or actions in column B. The steps are:

	A	B	C	D
1	Запустить приложение			
2	Создать новый файл			
3	Ввести текст			
4	Сохранить файл			
5	Распечатать файл			
6	Открыть существующий файл			
7	Модифицировать файл			
8	Выйти из приложения			
9				
10				

ПРИМЕР РАЗРАБОТКИ ТЕСТОВ

Microsoft Excel - Шаблон для разработки тестов v2						
File Edit View Insert Format Tools Data Window Help Adobe PDF						
A6 fx Arial 10 B I U						
	A	B	C	D	E	F
1	"Notepad" Функциональные тест-кейсы					
2	Тестировал(и):					
3	Дата(даты) тестирования:			ОС:		
4						
5	Идентификатор	Ссылка на требование	Модуль	Подмодуль/экран	Описание теста	Ожидаемый результат
6					Запустить приложение	
7					Создать новый файл	
8					Ввести текст	
9					Сохранить файл	
10					Распечатать файл	
11					Открыть существующий файл	
12					Модифицировать файл	
13					Выйти из приложения	
14						
15						

ПРИМЕР РАЗРАБОТКИ ТЕСТОВ

Microsoft Excel - Шаблон для разработки тестов v2							
Type a question for help							
	A	B	C	D	E	F	G
4	Идентификатор	Ссылка на требование	Модуль	Подмодуль/экран	Описание теста	Ожидаемый результат	Статус ("не тестировано", "выполнено успешно", "выполнение завершилось ошибкой")
5	ST_001	R1	Приложение не запущено		Запустить приложение 1. Выполнить команду notepad из командной строки	1. Появляется окно notepad с пустым файлом	Не тестировано
6	ST_002	R1, R16	Приложение		Создать новый файл 1. Выполнить последовательность команд "Файл" -> "Создать" с использованием меню	1. Создаётся новый файл (в рабочей области приложения пусто)	Не тестировано
7	ST_003	R34, R75.7	Приложение		Ввести текст 1. Набрать несколько слов 2. Удалить несколько слов	1. В рабочей области приложения отображается набранный текст 2. Удаляемые слова пропадают из рабочей области приложения	Не тестировано
8	ST_004	R23	Приложение	Работа с файлами	Сохранить файл 1. Создать новый файл. Ввести немного текста. 2. Выполнить последовательность команд "Файл" -> "Сохранить" с использованием меню 3. Выбрать каталог для сохранения файла и ввести имя файла 4. Нажать кнопку "Сохранить"	1. Создаётся новый файл, введённый текст отображается в рабочей области приложения 2. Появляется диалоговое окно "Сохранить файл" Какой каталог для сохранения должен отображаться по умолчанию? 3. Имя файла отображается в строке ввода 4. Диалоговое окно "Сохранить файл" исчезает, на диске в указанном каталоге появляется сохранённый файл	Не тестировано
9	ST_005	R45, R57, R92	Приложение	Работа с файлами	Распечатать файл 1. Выполнить последовательность команд "Файл" -> "Печать" с использованием меню 2. Следовать инструкциям	1. Открывается диалог "Печать документа" Какова реакция приложения на отсутствие в системе установленных принтеров?	Не тестировано
Титульная страница Smoke test Critical Path test							
Ready							

ТЕСТИРОВАНИЕ «ЧЕРНОГО ЯЩИКА»

- Функционал системы. (Делает ли программа то, что она должна делать согласно спецификации?)
- Контроль вводимых данных. (Как реагирует программа на некорректный ввод данных?)
- Выходные данные. (Правильно ли программа реагирует на рутинные действия пользователя?)



ТЕСТИРОВАНИЕ «ЧЕРНОГО ЯЩИКА»

ТЕХНИКИ ТЕСТИРОВАНИЯ:

эквивалентное разбиение;

анализ граничных значений;

анализ причинно-следственных связей;

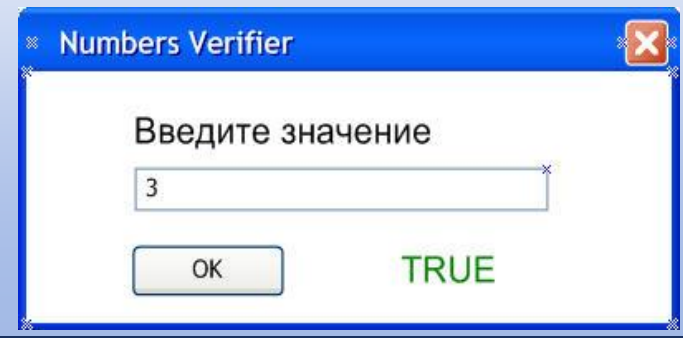
попарное тестирование;

предположение об ошибке.

МЕТОД ЭКВИВАЛЕНТНОГО РАЗБИЕНИЯ

- **Эквивалентный класс** — это одно или больше значений ввода, к которым программа применяет одинаковую логику.

- Форма валидации вводимого значения
- Если введено целочисленное значение от 1 до 99 (включительно), выводится сообщение «TRUE», иначе выдается сообщение об ошибке.



МЕТОД ЭКВИВАЛЕНТНОГО РАЗБИЕНИЯ: ПРИМЕР

Классы эквивалентности :

1. Любое целое в диапазоне от 1 до 99.
Как правило, вводится середина числового отрезка. *Позитивный тест.*
2. Любое число меньше 1. *Негативный тест.*
3. Любое число больше 99. *Негативный тест.*
4. Вещественное число и нечисловые значения (буквы, спецсимволы). *Негативный тест.*

МЕТОД ЭКВИВАЛЕНТНОГО РАЗБИЕНИЯ: ПРИМЕР

Тесты:

1. Ввести 1, 99, 50
2. Ввести 0
3. Ввести 100
4. Ввести 55.5, букву, спецсимвол:
~`!"@'#\$;%:^&?*()[]{}.,\/+=-_

АНАЛИЗ ГРАНИЧНЫХ ЗНАЧЕНИЙ

На границах классов эквивалентности меняется поведение системы.

Граничные условия — это ситуации, возникающие на высших и нижних границах входных классов эквивалентности.



АНАЛИЗ ГРАНИЧНЫХ ЗНАЧЕНИЙ

Этапы применения:

- В таблице перечисляются все переменные (входные и выходные).
- Для каждой переменной определяется разбиение на классы.
- Строятся все возможные комбинации классов.
- В качестве представителей классов берутся тесты на граничные, приграничные и (или) специальные значения.

АНАЛИЗ ГРАНИЧНЫХ ЗНАЧЕНИЙ : ПРИМЕР

- Программа предназначена для сложения двух целых чисел.
- Каждое из слагаемых – целое число от -99 до 99.
- Программа запрашивает у пользователя два числа, после чего выводит результат.

АНАЛИЗ ГРАНИЧНЫХ ЗНАЧЕНИЙ : ПРИМЕР

$2 + 7$

$4 + 7$

$5 + 7$

$3 + 8$

$3 + 6$

$3 + 5$

$3 + 3$

$7 + 3$

$7 + 7$

$2 + 2$

$6 + 4$

$2 + 5$

$(-3) + 7$

$(-3) + (-7)$

$3 + (-7)$

$0 + 7$

$3 + 0$

$0 + 0$

$99 + 7$

$3 + (-99)$

$(-99) + 0$

$99 + 99$

$99 + (-99)$

$(-99) + 99$

$100 + 3$

$100 + 100$

$100 + (-100)$

Избыточные
комбинации

Какие
комбинации
добавить



АНАЛИЗ ГРАНИЧНЫХ ЗНАЧЕНИЙ: ПРИМЕР

	Классы корректных данных (позитивные тесты)	Классы некорректных данных (негативные тесты)	Граничные и специальные значения
Первое слагаемое	от -99 до -10 от -9 до -1 0 от 1 до 9 от 10 до 99	> 99 < -99	0, 1, -1, 9, -9 10, -10 99, -99 100, -100
Второе	_"_"	_"_"	_"_"
Сумма	от -198 до -100 от -99 до -1 0 от 1 до 99 от 100 до 198	> 198 < -198	(-99, -99) (-49, -51) (99, 99) (49, 51)

МЕТОДЫ «ЧЕРНОГО ЯЩИКА»

- Размер файла (особенно кратный чему-либо).
- Объём памяти (особенно кратный чему-либо).
- Размер экрана, разрешение экрана.
- Размер окна.
- Количество цветов. Цветовую гамму.
- Версии операционной системы.
- Версии библиотек.
- Время (в т.ч. между событиями).
- Скорость передачи данных.
- Объём передаваемых данных.
- Интенсивность передачи данных.
- Переключение между разными алгоритмами (количество, время, скорость).

АНАЛИЗ ПРИЧИННО-СЛЕДСТВЕННЫХ СВЯЗЕЙ

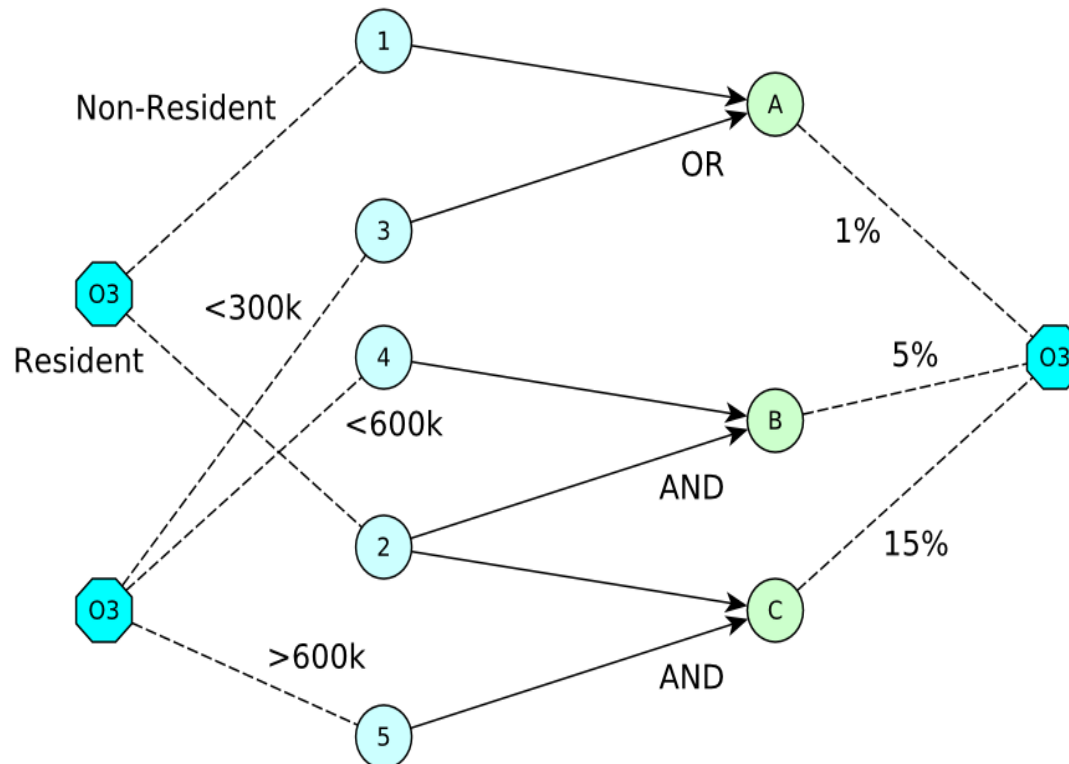
- Метод использует алгебру логики и оперирует понятиями «причина» и «следствие»:
 - причина – отдельное входное условие или класс эквивалентности.
 - следствие – выходное условие или преобразование системы.
- Этапы применения:
 1. В спецификации определяют множество причин и следствий.
 2. На основе анализа семантического (смыслового) содержания спецификации строят граф и (или) таблицу истинности, в которых каждой возможной комбинации причин (or, and) ставится в соответствие следствие.

АНАЛИЗ ПРИЧИННО-СЛЕДСТВЕННЫХ СВЯЗЕЙ: ПРИМЕР

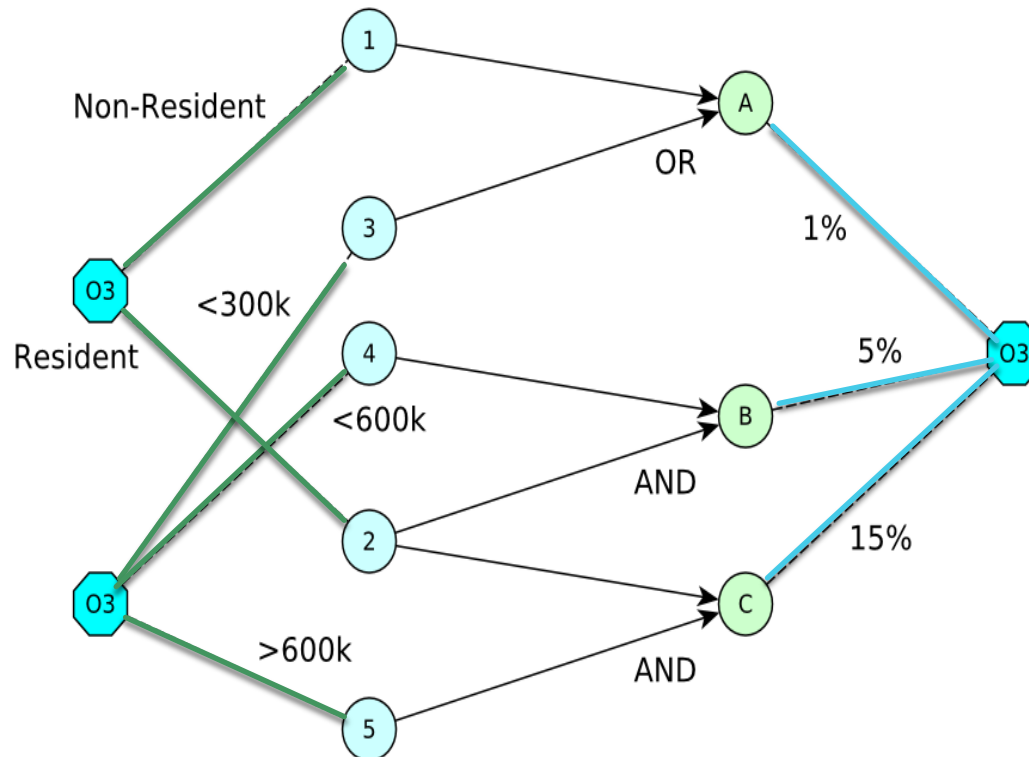
Расчет городского налога:

- Нерезиденты платят 1% от общего дохода
- Резиденты платят
 - 1% от дохода, если он не превышает 300,000 в год
 - 5% от дохода, если он не превышает 600,000 в год
 - 15% от дохода, если он превышает 600,000 в год

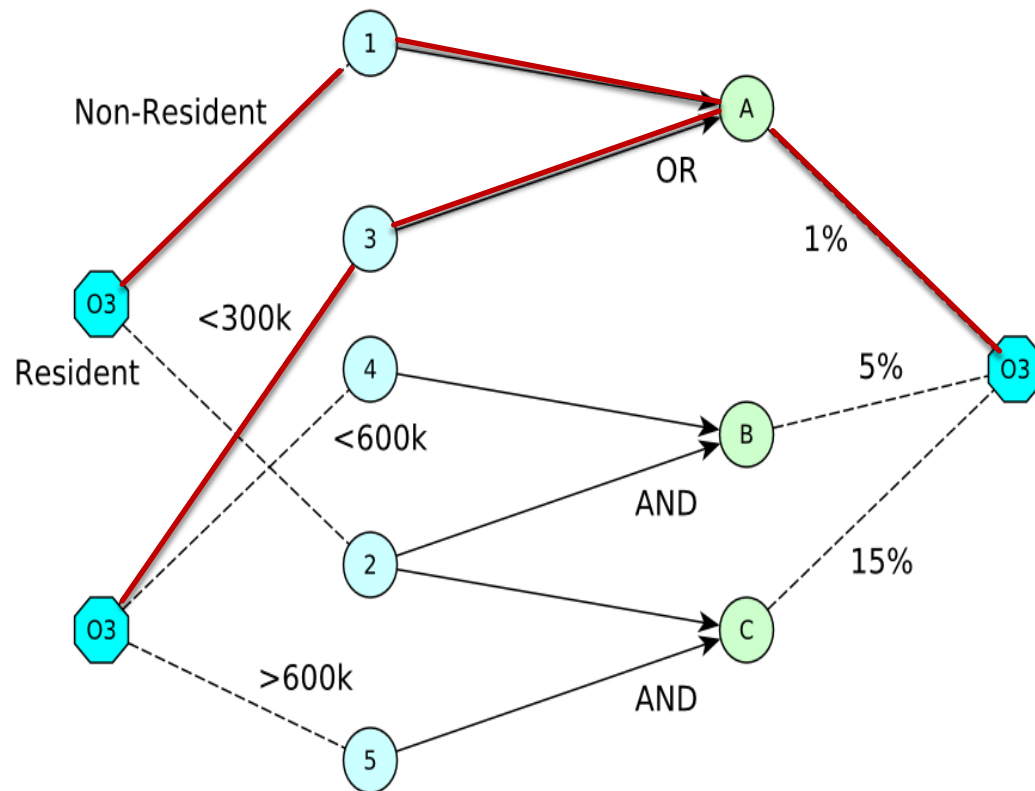
АНАЛИЗ ПРИЧИННО-СЛЕДСТВЕННЫХ СВЯЗЕЙ: ПРИМЕР



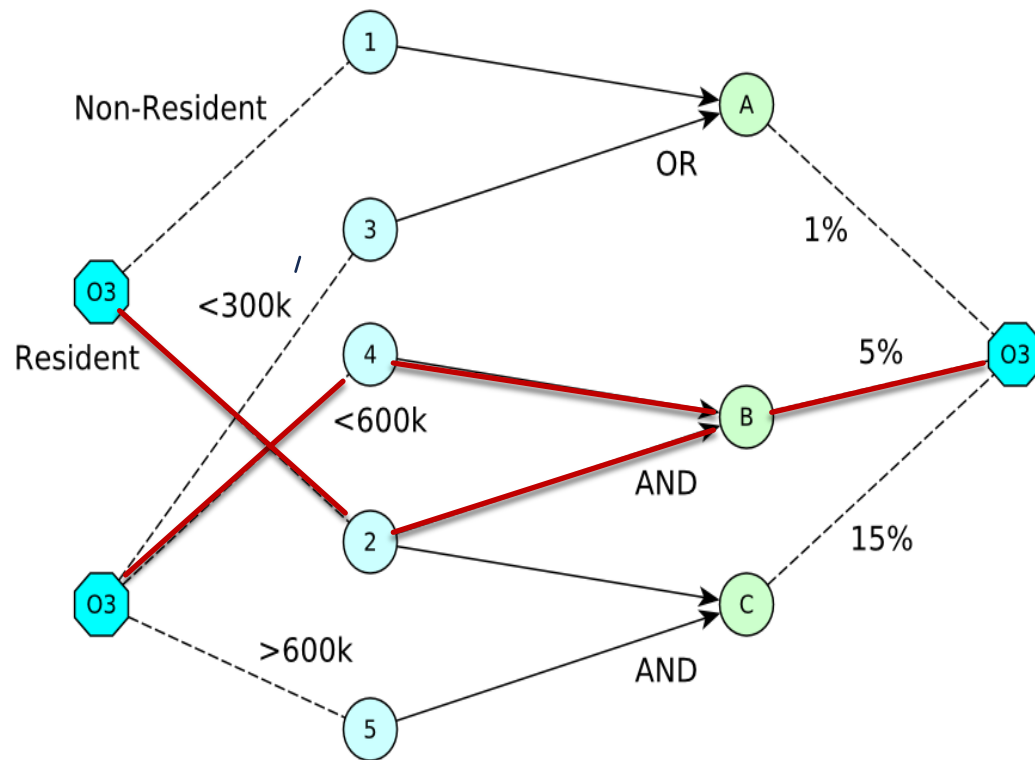
АНАЛИЗ ПРИЧИННО-СЛЕДСТВЕННЫХ СВЯЗЕЙ: ПРИМЕР



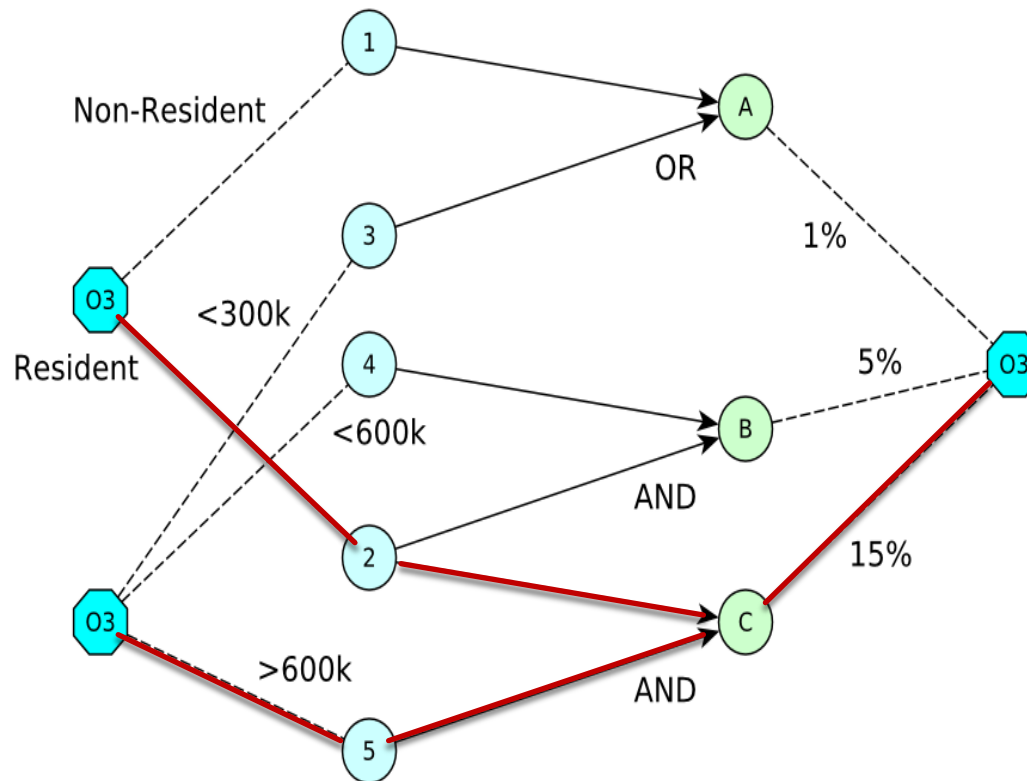
АНАЛИЗ ПРИЧИННО-СЛЕДСТВЕННЫХ СВЯЗЕЙ: ПРИМЕР



АНАЛИЗ ПРИЧИННО-СЛЕДСТВЕННЫХ СВЯЗЕЙ: ПРИМЕР



АНАЛИЗ ПРИЧИННО-СЛЕДСТВЕННЫХ СВЯЗЕЙ: ПРИМЕР



МЕТОД ПОПАРНОГО ТЕСТИРОВАНИЯ

- Метод формирования наборов тестовых данных, в которых каждое тестируемое значение каждого из проверяемых параметров хотя бы единожды сочетается с каждым тестируемым значением всех остальных проверяемых параметров.
- Идея метода – параметры попарно зависимы.

МЕТОД ПОПАРНОГО ТЕСТИРОВАНИЯ: ПРИМЕР

Параметр 1	Параметр 2	Параметр 3
Значение 1.1	Значение 2.1	Значение 3.1
Значение 1.2	Значение 2.2	Значение 3.2

#	Параметр 1	Параметр 2	Параметр 3
1	Значение 1.1	Значение 2.1	Значение 3.1
2	Значение 1.1	Значение 2.2	Значение 3.1
3	Значение 1.2	Значение 2.1	Значение 3.1
4	Значение 1.2	Значение 2.2	Значение 3.1
5	Значение 1.1	Значение 2.1	Значение 3.1
6	Значение 1.1	Значение 2.1	Значение 3.2
7	Значение 1.2	Значение 2.1	Значение 3.1
8	Значение 1.2	Значение 2.1	Значение 3.2
9	Значение 1.1	Значение 2.1	Значение 3.1
10	Значение 1.1	Значение 2.1	Значение 3.1
11	Значение 1.1	Значение 2.2	Значение 3.1
12	Значение 1.1	Значение 2.2	Значение 3.2

0	0	0
0	1	0
1	0	0
1	1	0
0	0	0
0	0	1
1	0	0
1	0	1
0	0	0
0	0	1
0	1	0
0	1	1

МЕТОД ПОПАРНОГО ТЕСТИРОВАНИЯ: ПРИМЕР

#	Параметр 1	Параметр 2	Параметр 3
1	Значение 1.1	Значение 2.1	Значение 3.1
2	Значение 1.1	Значение 2.2	Значение 3.1
3	Значение 1.2	Значение 2.1	Значение 3.1
4	Значение 1.2	Значение 2.2	Значение 3.1
5	Значение 1.1	Значение 2.1	Значение 3.2
6	Значение 1.2	Значение 2.1	Значение 3.2
7	Значение 1.1	Значение 2.2	Значение 3.2

0	0	0
0	1	0
1	0	0
1	1	0
0	0	1
1	0	1
0	1	1

#	Параметр 1	Параметр 2	Параметр 3
1	Значение 1.1	Значение 2.1	Значение 3.1
2	Значение 1.1	Значение 2.2	Значение 3.2
3	Значение 1.2	Значение 2.1	Значение 3.2
4	Значение 1.2	Значение 2.2	Значение 3.1

0	0	0
0	1	1
1	0	1
1	1	0

ПРЕДПОЛОЖЕНИЕ ОБ ОШИБКЕ

- Самый неформальный метод тестирования
- Основан на интуиции

ТЕСТИРОВАНИЕ «БЕЛОГО ЯЩИКА». ПРИНЦИПЫ РАЗРАБОТКИ ТЕСТОВ

Болтовня ничего не стоит. Покажите
мне код.

Линус Торвальдс

При достаточном количестве
глаз баги всплывают на поверхность
Закон Линуса



ЛИНУС ТОРВАЛЬДС

28 декабря 1969 года Хельсинки, Финляндия



Линус Торвальдс , шведско-американский программист

Будучи студентом, разработал ядро Linux (1991 г.) и файловые системы ext (Extended — расширение для файловой системы Minix), и ext2.

2014 —медаль «Пионер компьютерной техники» (IEEE).

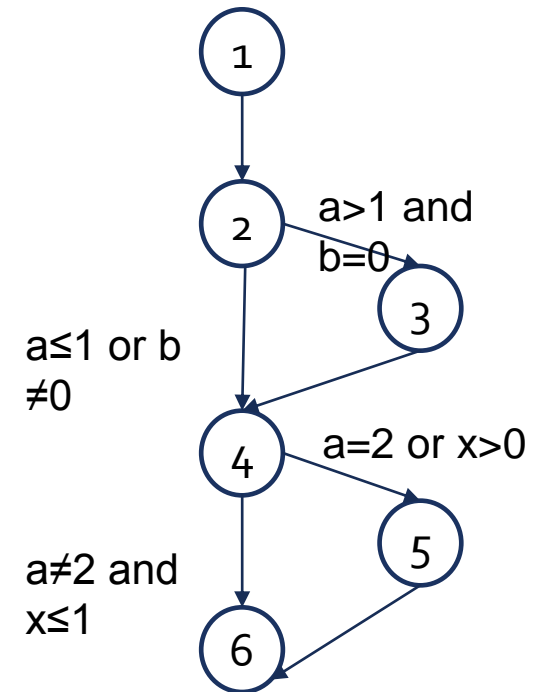
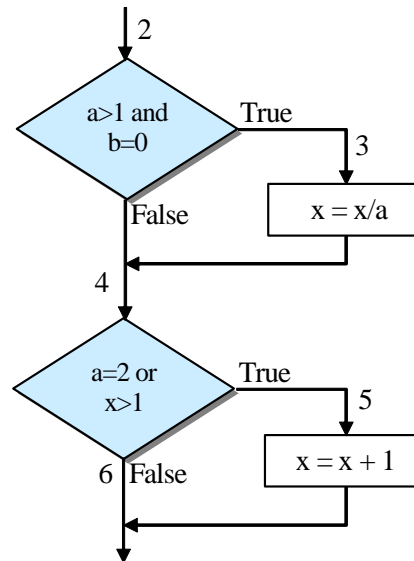
ПРИНЦИПЫ СОЗДАНИЯ ТЕСТОВ

- На основе анализа **потока управления** – управляющего графа программы (УГП, *Control Flow Graf*) $G(V, A)$, где $V(V_1, \dots, V_m)$ – множеств вершин (операторов), $A(A_1, \dots, A_n)$ – множество дуг (управлений), соединяющих операторы-вершины.
- **Путь** – последовательность вершин и дуг УГП, в которой любая дуга выходит из вершины V_i и приходит в вершину V_j .
- **Ветвь** – путь (V_1, V_2, \dots, V_k) , где V_1 – либо первый, либо условный оператор программы; V_k – либо условный оператор, либо оператор выхода из программы, а все остальные операторы – безусловные.

УПРАВЛЯЮЩИЙ ГРАФ ПРОГРАММЫ: ПРИМЕР

Void m (float a, float b, float x)

1. {
2. If (a>1 && b==0)
3. x=/[a](#);
4. If (a==2||x>1)
5. x++;
6. }



Пути: (1-2-3-4-5-6); (1-2-4-6); (1-2-4-5-6); (1-2-3-4-6)
Ветви: (2-3-4); (4-5-6); (2-4); (4-6)

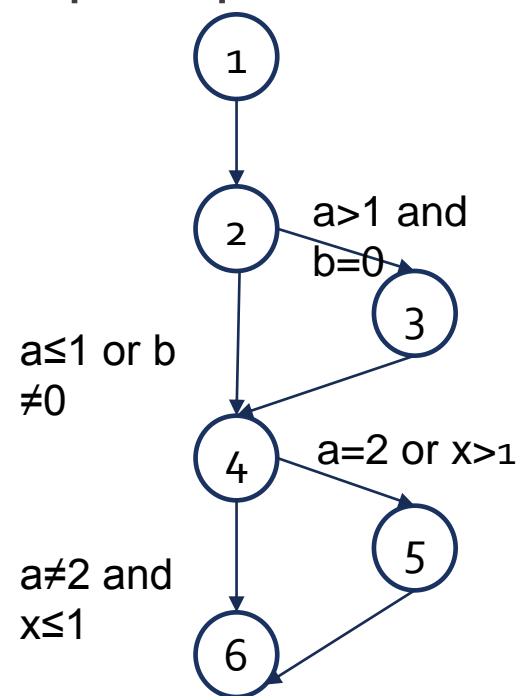
ТЕСТИРОВАНИЕ «БЕЛОГО ЯЩИКА»: МЕТОДЫ

- **Тестирование операторов** (критерий покрытия **CO**) – набор тестов в совокупности должен обеспечить прохождение каждого оператора не менее одного раза.

$a = 2, b = 0, x = 3.$

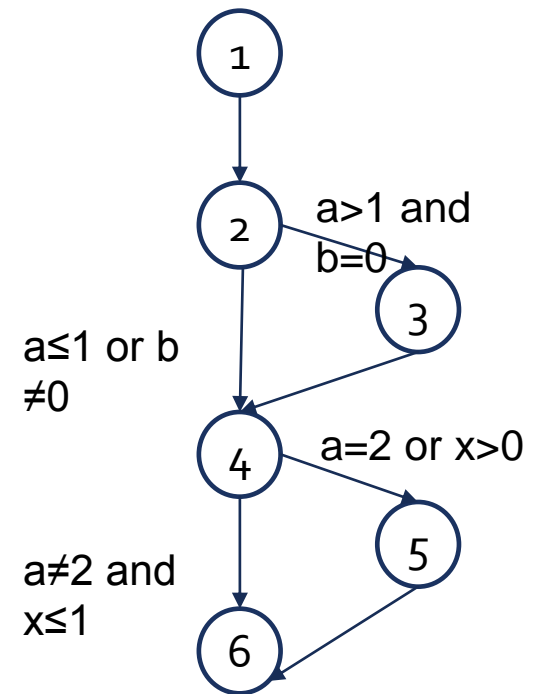
Существует путь (1-2-4-6), в котором значение x вообще не меняется и, если здесь есть ошибка, она не будет обнаружена

Критерий необходимый,
но не является достаточным!!!



СТРАТЕГИЯ «БЕЛОГО ЯЩИКА»: МЕТОДЫ

- **Тестирование решений или ветвей**
(критерий покрытия **C₁**) – набор тестов в совокупности должен обеспечить прохождение каждой ветви не менее одного раза, то есть каждое условие в программе должно принять как истинное, так и ложное значение.



СТРАТЕГИЯ «БЕЛОГО ЯЩИКА»: МЕТОДЫ

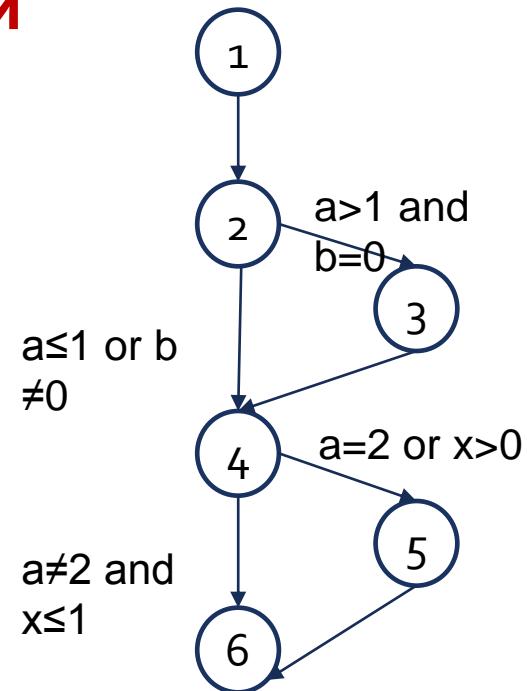
■ Тестирование решений или ветвей

1 вариант

- $a = 2, b = 0, x = 3$ – путь (1-2-3-4-5-6)
- $a = 3, b = 1, x = 1$ – путь (1-2-4-6).

2 вариант

- $a = 3, b = 0, x = 1$ – путь (1-2-3-4-6);
- $a = 3, b = 1, x = 3$ – путь (1-2-4-5-6).



Критерий удовлетворяет требованию достаточности, но не удовлетворяет требованию полноты!!!

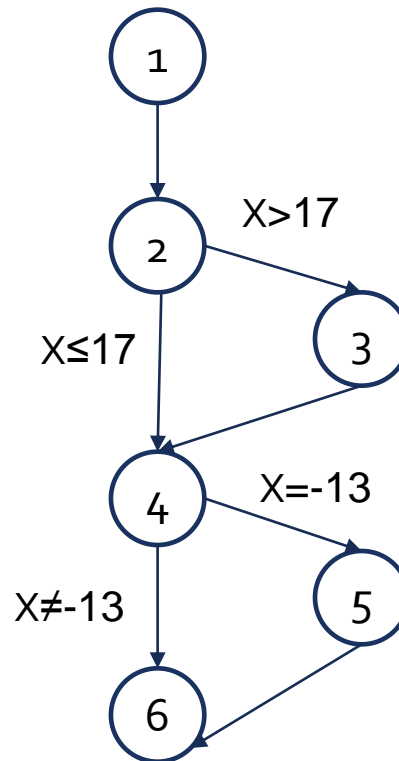
СТРАТЕГИЯ «БЕЛОГО ЯЩИКА»: МЕТОДЫ

- **Тестирование путей** (критерий покрытия **C2**) - набор тестов в совокупности должен обеспечить прохождение каждого пути не менее 1 раза. Если программа содержит цикл (в особенности с неявно заданным числом итераций), то число итераций ограничивается константой (как правило, $const=2$).

СТРАТЕГИЯ «БЕЛОГО ЯЩИКА»: ПРИМЕР

```
void Method (int x)
```

```
1  {  
2  if (x>17)  
3      x = 17-x;  
4  if (x== -13)  
5      x = 0;  
6  }
```



- Покрытие операторов
 $x = 30$
- Покрытие ветвей
 $x = 30, x = 17$
- Покрытие путей
 $x = 30, x = 17,$
 $x = -13, x = 18$

СТРАТЕГИЯ «БЕЛОГО ЯЩИКА»: МЕТОДЫ

■ Тестирование путей

Void m (float a, float b, float x)

1. {
2. If (**a>1** && **b==0**)
3. x=/a;
4. If (**a==2**||**x>1**)
5. x++;
6. }

Тест удовлетворяет всем требованиям!!!

1	2	3	4
1	1	1	1
0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
.....

ТЕСТИРОВАНИЕ «БЕЛОГО ЯЩИКА»: МЕТОДЫ

- **Покрывтие условий:** возможные результаты каждого условия в решении должны быть выполнены, по крайней мере, один раз.

ПОКРЫТИЕ УСЛОВИЙ: ПРИМЕР

■ Условия:

1) $a > 1$; 2) $b = 0$; 3) $a = 2$; 4) $x > 1$.

■ Необходимо реализовать все возможные ситуации:

1) $a > 1, a \leq 1$; 2) $b = 0, b \neq 0$; 3) $a = 2, a \neq 2$; 4) $x > 1, x \leq 1$.

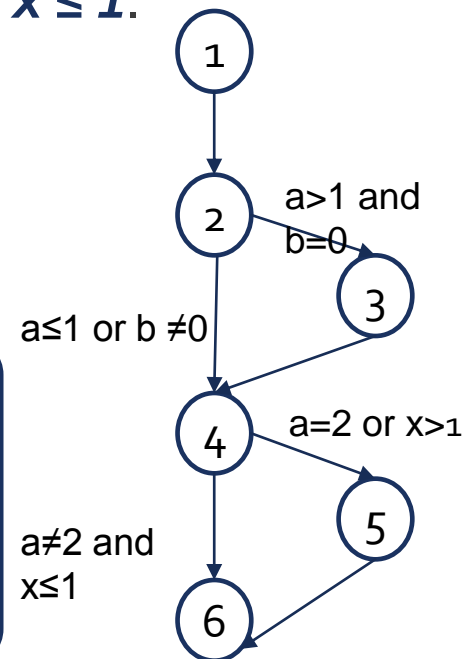
■ Тесты, удовлетворяющие покрытию условий:

**1. $a = 2, b = 0,$
 $x = 3$**

- путь 1-2-3-4-5-6,
условия:
1) – да; 2) – да;
3) – да; 4) – да.

**2. $a = 1, b = 1,$
 $x = 1$**

- путь 1-2-4-6,
условия:
1) – нет; 2) – нет;
3) – нет; 4) – нет.



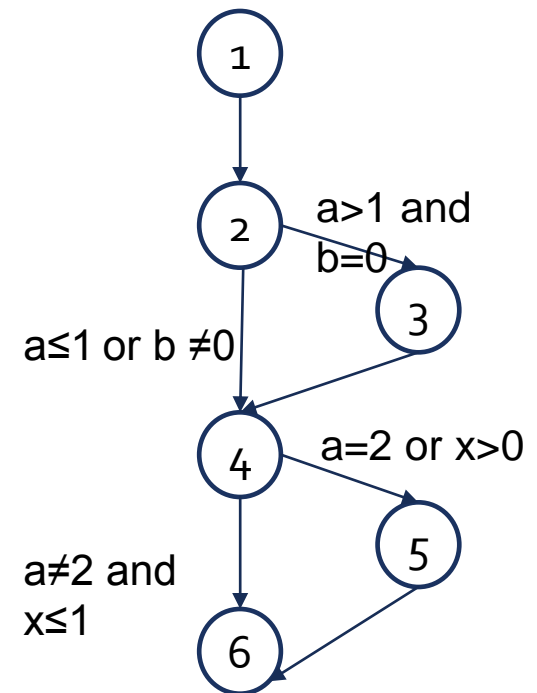
ТЕСТИРОВАНИЕ «БЕЛОГО ЯЩИКА»: МЕТОДЫ

- **Комбинаторное покрытие условий:** все возможные комбинации результатов условий в каждом решении, а также каждый оператор должен выполняться, по крайней мере, один раз.

КОМБИНАТОРНОЕ ПОКРЫТИЕ УСЛОВИЙ

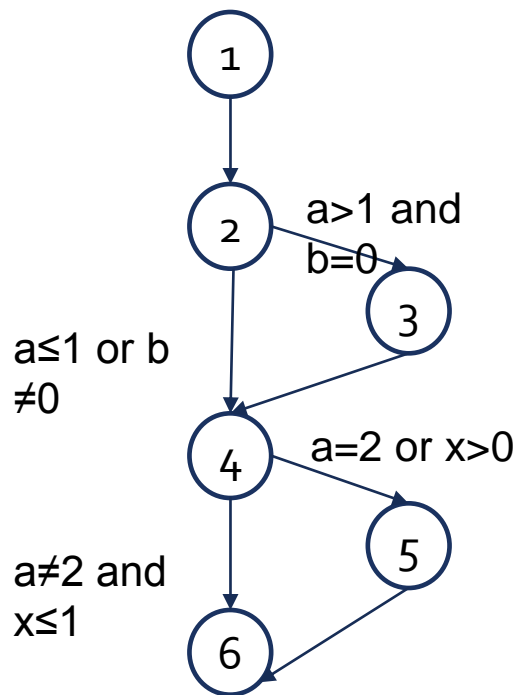
■ Комбинации условий:

1) $a > 1, b = 0.$	2) $a > 1, b \neq 0.$
3) $a \leq 1, b = 0.$	4) $a \leq 1, b \neq 0.$
5) $a = 2, x > 1.$	6) $a = 2, x \leq 1.$
7) $a \neq 2, x > 1.$	8) $a \neq 2, x \leq 1.$



КОМБИНАТОРНОЕ ПОКРЫТИЕ УСЛОВИЙ

■ Комбинации условий:



$a = 2, b = 0, x = 4$ —
проверяет
комбинации
(1), (5);

$a = 2, b = 1, x = 1$ —
проверяет
комбинации
(2), (6);

$a = 1, b = 0, x = 2$ —
проверяет
комбинации
(3), (7);

$a = 1, b = 1, x = 1$ —
проверяет
комбинации
(4), (8).

ТЕСТИРОВАНИЕ «БЕЛОГО ЯЩИКА»: МЕТОДЫ

- **Покрывтие условий/решений**, совмещающий требования по покрытию и решений, и условий: возможные результаты каждого условия должны выполняться, по крайней мере, один раз и каждой точке управление должно передаваться, по крайней мере, один раз.

СТРАТЕГИИ ТЕСТИРОВАНИЯ: СРАВНЕНИЕ

Критерий	Black Box	White Box
<i>Определение</i>	тестирование, как функциональное, так и нефункциональное, не предполагающее знания внутреннего устройства компонента или системы	тестирование, основанное на анализе внутренней структуры компонента или системы
<i>Уровни, к которым применима</i>	В основном: <ul style="list-style-type: none">• Приемочное тестирование• Системное тестирование	В основном: <ul style="list-style-type: none">• Юнит-тестирование• Интеграционное тестирование
<i>Кто выполняет</i>	Как правило, тестировщики	Как правило, разработчики
<i>Знание программирования</i>	Не нужно	Необходимо
<i>Знание реализации</i>	Не нужно	Необходимо
<i>Основа для тест-кейсов</i>	Спецификация, требования	Проектная документация

ВЫВОДЫ:

Стратегии «белого ящика» и «черного ящика»:

- определяются в зависимости от доступа к исходному коду и знанию внутренней структуры программы;
- используют различные методы и критерии тестирования;
- направлены на обнаружение максимального количества ошибок при минимальном количестве тестов;
- обнаруживают различные классы ошибок \Rightarrow не являются альтернативной, а дополняют друг друга.

ДОМАШНЕЕ ЗАДАНИЕ:

- Сравнить методы покрытия решений и покрытия условий
- Сравнить методы покрытия путей и комбинаторного покрытия условий

ЗАКЛЮЧЕНИЕ

Не волнуйтесь, если что-то не работает. Если бы всё работало, вас бы уволили.

Mosher's Law of Software Engineering