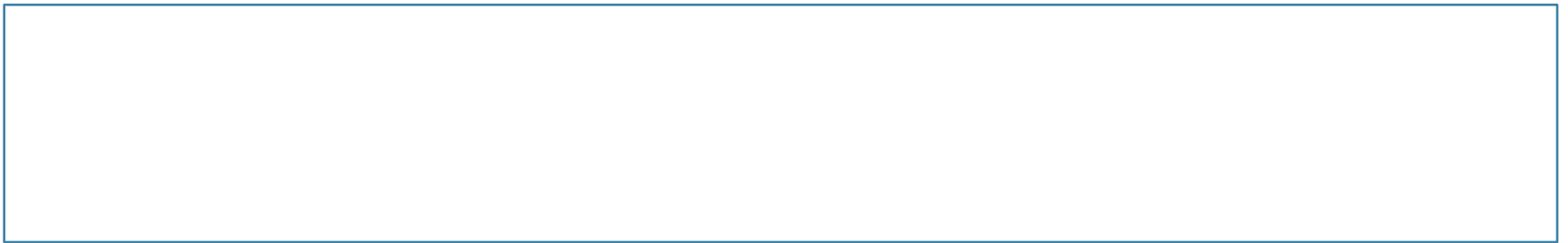




# СТРАТЕГИИ ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ



# ОПРЕДЕЛЕНИЯ

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

- Методы тестирования \*
- Критерии тестирования

\* – инструментальные средства в случае автоматизированного тестирования

# СТРАТЕГИЯ «ЧЕРНОГО ЯЩИКА»

- Стратегия «черного ящика» (*black-box testing*) — стратегия 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

# СТРАТЕГИЯ «ЧЕРНОГО ЯЩИКА»: КЛАССИФИКАЦИЯ

## Функциональное тестирование (*functional testing*)

- вид тестирования, основанный на анализе функциональных спецификаций

## Нефункциональное тестирование (*non-functional testing*)

- вид тестирования, направленный на проверку реализации нефункциональных требований

# ИДЕАЛЬНЫЙ КРИТЕРИЙ ТЕСТИРОВАНИЯ: ТРЕБОВАНИЯ

## Достаточность

- Критерий должен показывать, когда некоторое конечное множество тестов достаточно для тестирования данной программы.

## Полнота

- В случае ошибки должен существовать тест из множества тестов, удовлетворяющих критерию, который раскрывает ошибку.

## Надежность

- Любые два множества тестов, удовлетворяющих критерию, одновременно должны (не) раскрывать ошибки программ.

## Проверяемость

- Например, вычисляемость на тестах.

# СТРАТЕГИЯ «ЧЕРНОГО ЯЩИКА»: ТЕСТИРУЕМЫЕ ЭЛЕМЕНТЫ

- 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?)



# СТРАТЕГИЯ «ЧЕРНОГО ЯЩИКА»:

Методы  
тестирования:

эквивалентное разбиение;

анализ граничных значений;

анализ причинно-следственных связей;

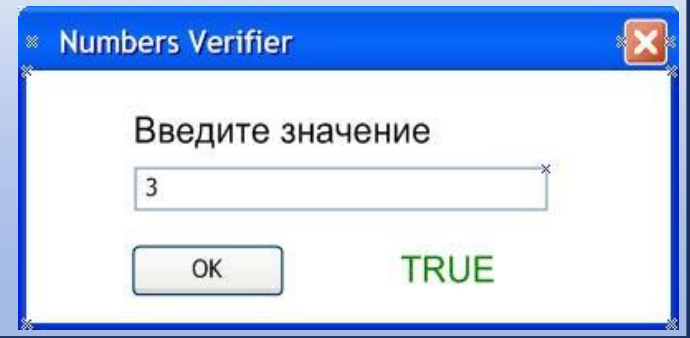
попарное тестирование;

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 – двузначное целое число.
- Программа запрашивает у пользователя два числа, после чего выводит результат.

# АНАЛИЗ ПРИЧИННО-СЛЕДСТВЕННЫХ СВЯЗЕЙ

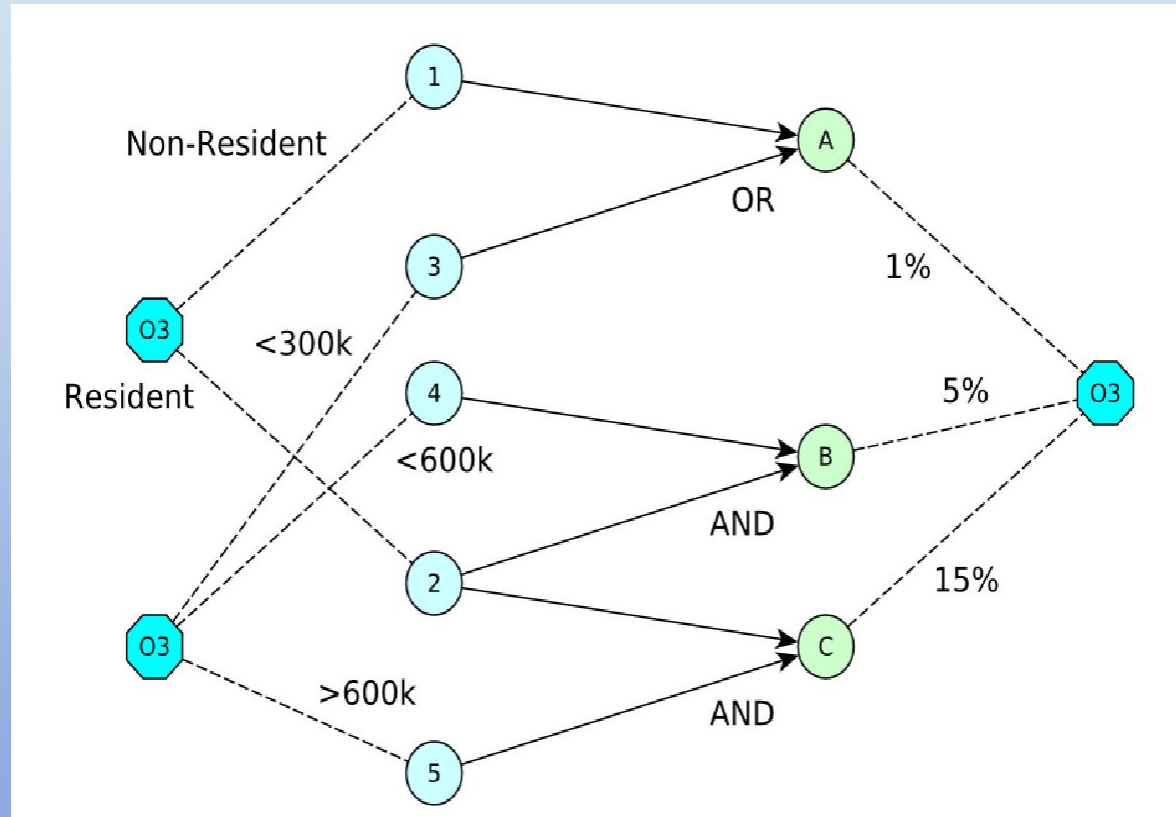
- Метод использует алгебру логики и оперирует понятиями «причина» и «следствие»:
  - причина – отдельное входное условие или класс эквивалентности.
  - следствие – выходное условие или преобразование системы.
- Этапы применения:
  1. В спецификации определяют множество причин и следствий.
  2. На основе 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





# 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

# ПРЕДПОЛОЖЕНИЕ ОБ ОШИБКЕ

- Самый неформальный метод тестирования
- Основан на интуиции

# ФУНКЦИОНАЛЬНЫЕ КРИТЕРИИ: ПРИМЕР

- **Покрытие 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.

# 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"><li>• Приемочное тестирование</li><li>• Системное тестирование</li></ul>	В основном: <ul style="list-style-type: none"><li>• Юнит-тестирование</li><li>• Интеграционное тестирование</li></ul>
<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