Составление наборов тестовых данных для структурного тестирования. Стратегия "белого ящика"

МДК.05.03

Тестирование по маршрутам

- При использовании стратегии «белого ящика» тестовые наборы формируют путем анализа маршрутов, предусмотренных алгоритмом. Под маршрутами при этом понимают последовательности операторов программы, которые выполняются при конкретном варианте исходных данных.

Последовательность составления тестов

1. На основе алгоритма (текста) программы формируется потоковый граф (или граф-схема).

2. Выбирается критерий тестирования.

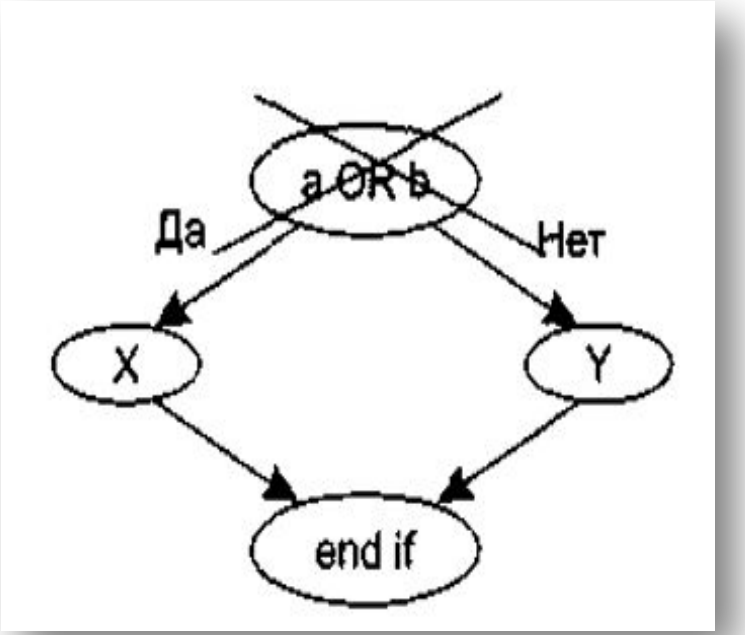
3. Подготавливаются тестовые варианты, инициирующие выполнение каждого пути. Каждый тестовый вариант формируется в следующем виде:

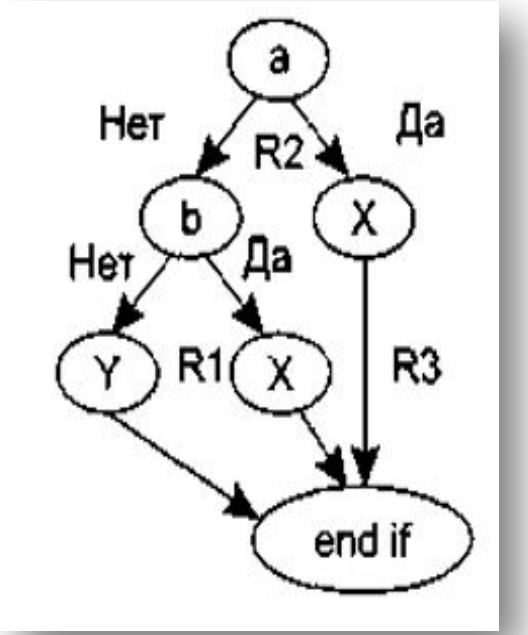
- Исходные данные
- Ожидаемые результаты

Потоковый граф

if a OR b then x else y end if.

Узлы (вершины) потокового графа соответствуют линейным участкам программы.
Дуги (ориентированные ребра) потокового графа отображают поток управления в программе (передачи управления между операторами).





Различают *операторные* и *предикатные* узлы.
Из операторного узла выходит одна дуга, а из предикатного – две дуги.
Предикатные узлы соответствуют простым условиям в программе.

Цикломатическая сложность

- Число независимых линейных путей в базовом множестве определяется цикломатической сложностью алгоритма, которая вычисляется одним из способов:

1) Цикломатическая сложность V(G) равна количеству регионов потокового графа; Регион - замкнутая область, образованная дугами и узлами. Окружающая граф среда рассматривается как дополнительный регион. *Например, показанный на предыдущем слайде граф имеет три региона – R1, R2, R3;*

2) Цикломатическая сложность определяется по формуле:

V(G) = E-N+2, где E – количество дуг, N – количество узлов потокового графа;

3) Цикломатическая сложность V(G) = p +1, где p – количество предикатных узлов G.

(Примечание Предлагается выполнять расчет по 2 или 3 способу)

Пример 1

- Цикломатическая сложность алгоритма на рисунке:

1) V(G) = 3 региона;

2) V(G) = 7 дуг - 6 узлов + 2 = 3;

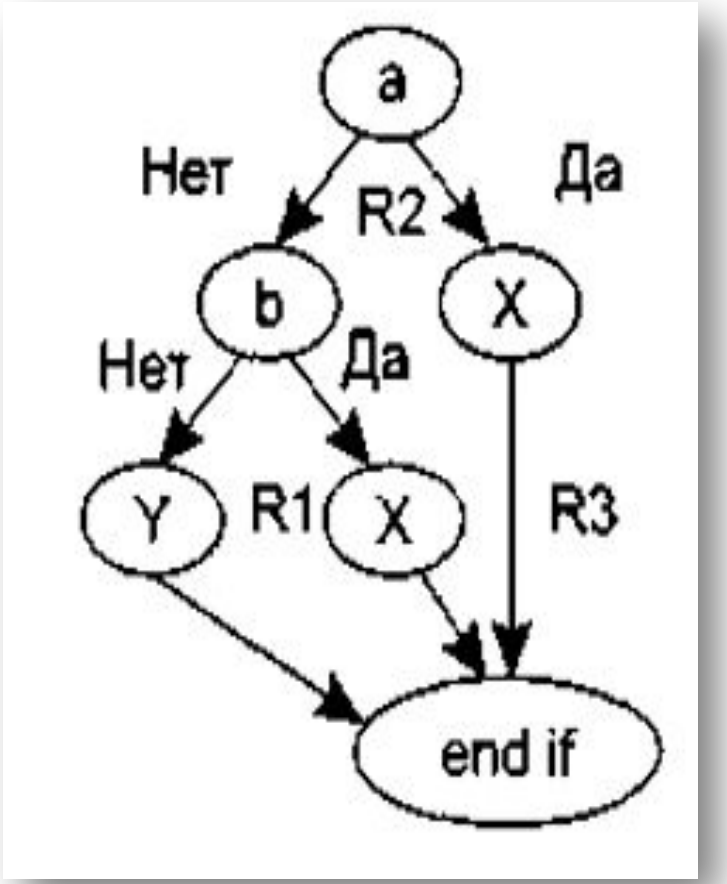
3) V(G) = 2 предикатных узла + 1 = 3.

- Независимые пути:

Путь 1: a - x – end if.

Путь 2: a - b - x – end if.

Путь 3: a - b - y – end if.



Критерии покрытия

1. Критерий покрытия операторов подразумевает такой подбор тестов, чтобы каждый оператор программы выполнялся, по крайней мере, один раз. Это необходимое, но недостаточное условие для приемлемого тестирования.

2. Критерий покрытия решений (переходов) должен иметь такое количество и состав тестов, чтобы результат проверки каждого условия принимал значения «истина» или «ложь», по крайней мере, один раз. Критерий покрытия решений удовлетворяет критерию покрытия операторов, но является более «сильным».

3. Критерий покрытия условий является еще более «сильным» по сравнению с предыдущими. В этом случае формируют некоторое количество тестов, достаточное для того, чтобы все возможные результаты каждого условия в решении были выполнены, по крайней мере, один раз. К этому критерию требуется дополнение, заключающееся в том, что каждой точке входа управление должно быть передано, по крайней мере, один раз.

Согласно покрытию решений/условий тесты должны составляться так, чтобы, по крайней мере, один раз выполнились все возможные результаты каждого условия и все результаты каждого решения, и каждому оператору управление передавалось, по крайней мере, один раз.

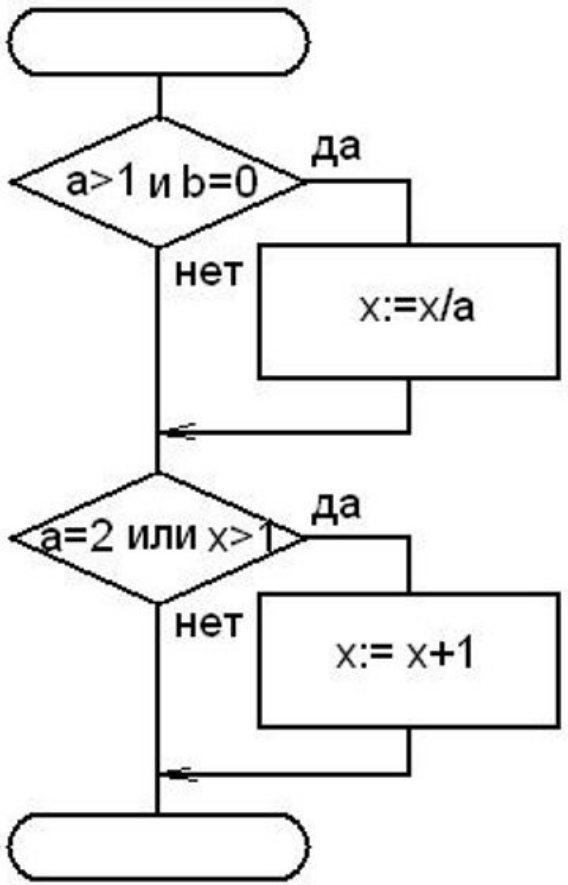
4. Критерий «Комбинаторное покрытие условий» требует создания такого множества тестов, чтобы все возможные комбинации результатов условий в каждом решении и все операторы выполнялись, по крайней мере, один раз.

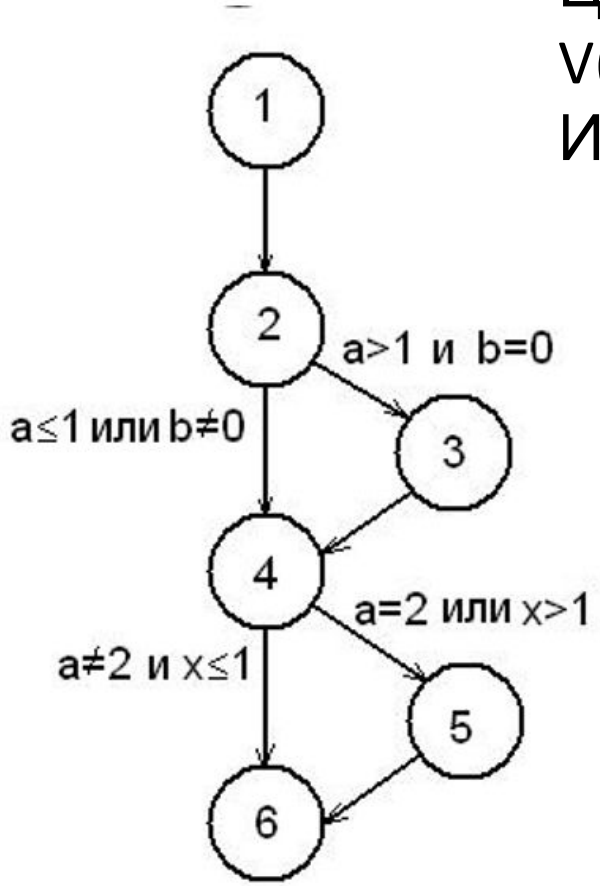
Пример 2

Требуется выполнить структурное тестирование текста программы, которая определяет значение x в зависимости от значений параметров процедуры.

```
Procedure m(a, b, x);
    begin
    if(a>1) and (b=0) then x:=x /a;
        if(a=2) or (x>1) then x:=x+1;
    end;
```

Схема алгоритма процедуры и граф передачи управления





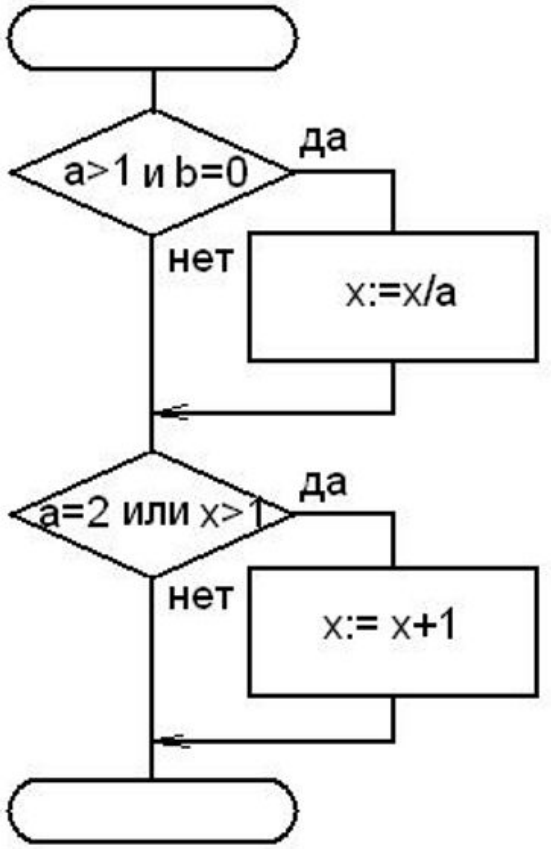
Цикломатическая сложность
V(G) = E-N+2 = 7 – 6 + 2= 3
Или V(G) = p +1 = 2 + 1 = 3

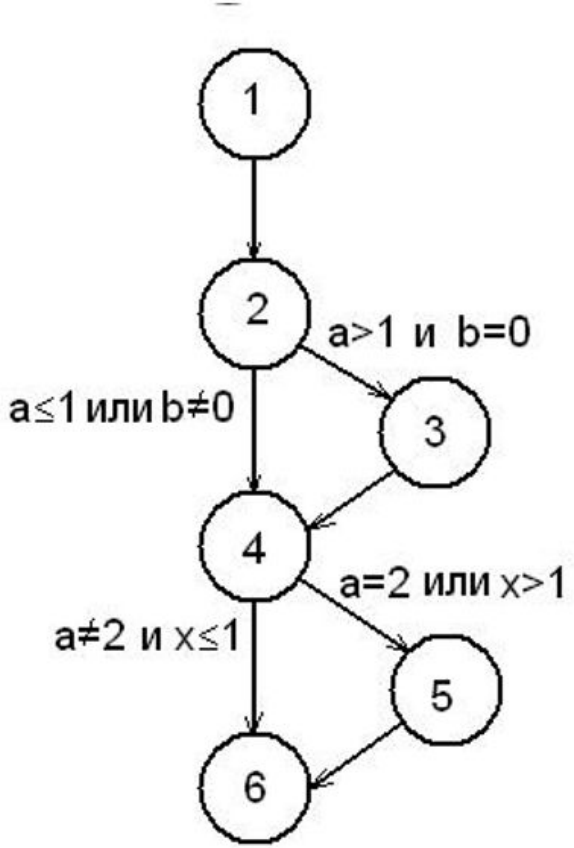
Акт

Покрытие операторов

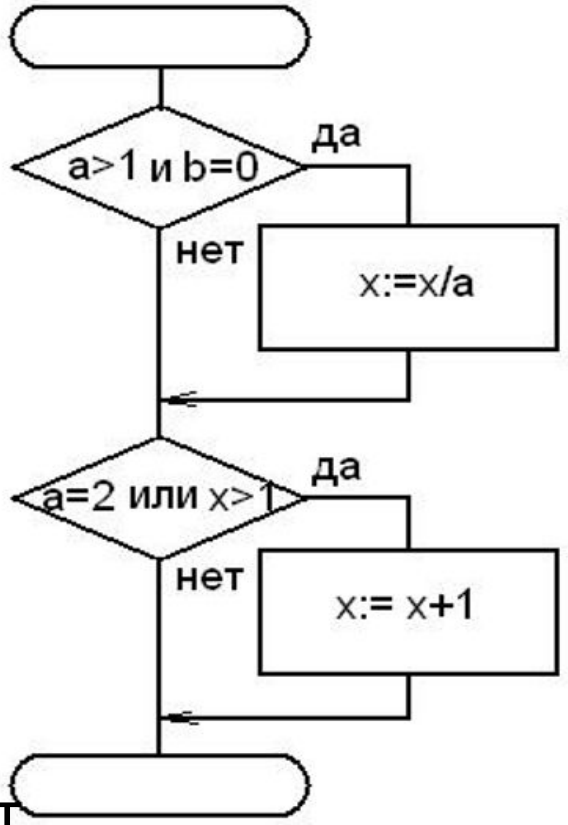
Покрытие операторов будет реализовано при a = 2, b = 0, x = 3. Однако для проверки программы этого недостаточно. *Например*, если при написании программы в первом условии указано, что (a > 1) or (b = 0), или, если во втором условии вместо x > 1 записано x > 0, то эти ошибки обнаружены не будут.

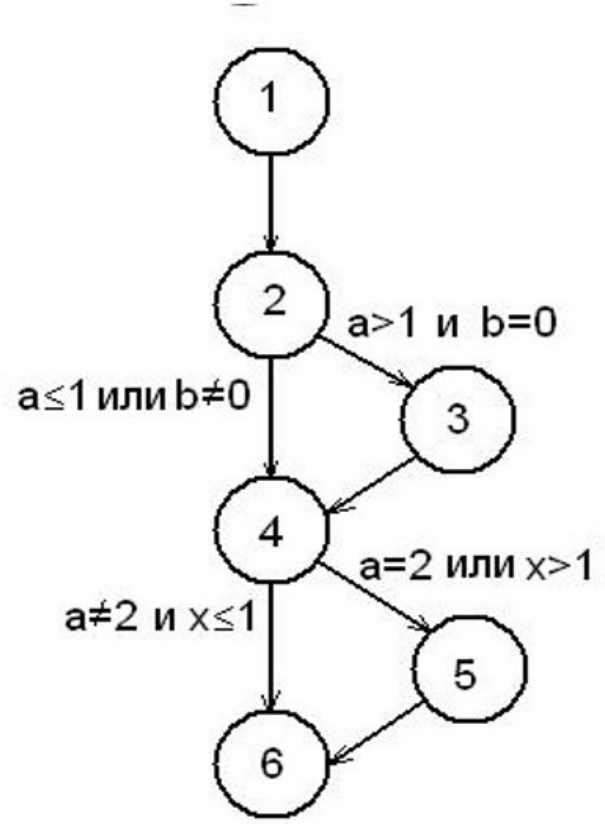
Также существует путь 1-2-4-6, в котором x вообще не меняется и, если здесь есть ошибка, она не будет обнаружена.





Покрытие решений





По методу покрытия решений рассматриваемую программу можно протестировать двумя тестами, покрывающими либо пути:
1-2-4-6, 1-2-3-4-5-6,
либо : 1-2-3-4-6, 1-2-4-5-6,
<u>например</u>:

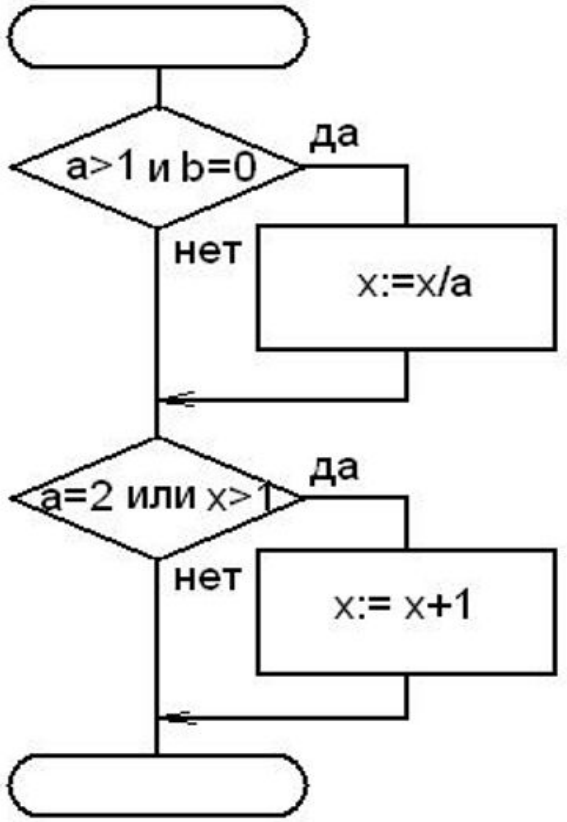
- a = 3, b = 0, x = 3 — путь 1-2-3-4-5-6;
- a = 2, b = 1 , x = 1 — путь 1-2-4-5-6.

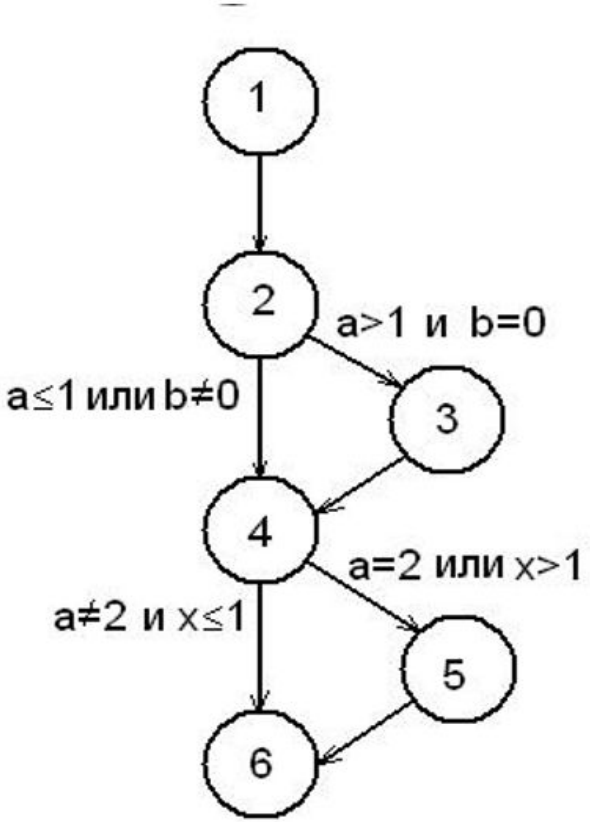
Однако путь, где x не меняется, будет проверен с вероятностью 50 %: если во втором условии вместо x > 1 записано x < 1, то этими двумя тестами ошибка обнаружена не будет.

Покрытие условий

Покрытие условий проверяет четыре условия:

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





Тесты, удовлетворяющие этому условию:

- a= 2, b = 0, x = 4 — путь 1-2-3-4-5-6, условия: 1-да, 2-да, 3-да, 4-да
- a = 1, b = 1, x = 1 — путь 1-2-4-6, условия: 1-нет, 2-нет, 3–нет, 4-нет.

- Критерий покрытия условий часто удовлетворяет критерию покрытия решений, но не всегда. Тесты критерия покрытия условий для рассмотренных примеров покрывают результаты всех решений, но это случайное совпадение.

Например, тесты:

- a=1, b=0, x=3 — путь 1-2-3-6, условия: 1-нет, 2-да, 3-нет, 4-да;
- a=2, b=1, x=1 — путь 1-2-3-4-5-6, условия: 1-да, 2-нет, 3-да, 4-нет

покрывают результаты всех условий, но только два из четырех результатов решений: не выполняется результат «истина» первого решения и результат «ложь» второго.

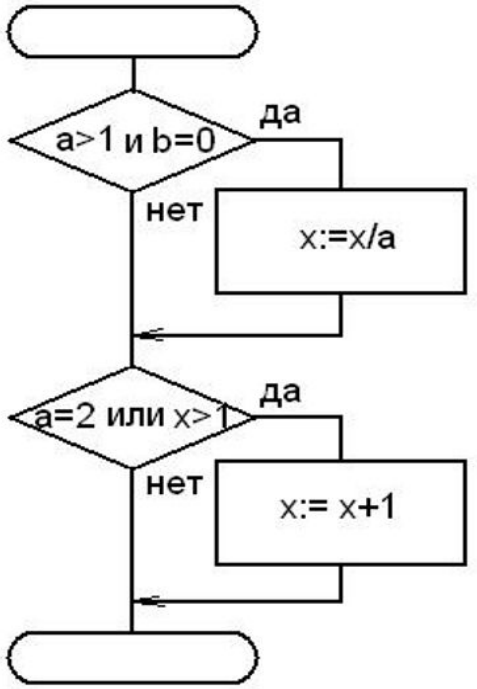
Основной недостаток метода – недостаточная чувствительность к ошибкам в логических выражениях.

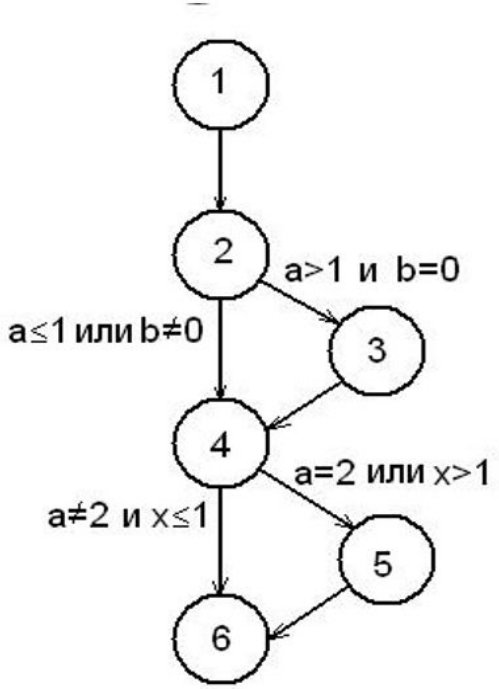
Покрытие решений/условий

этому критерию удовлетворяют тесты:

a=2, b=0, x=4 – путь 1-2-3-4-5-6, условия: 1-да, 2-да, 3-да, 4-да;

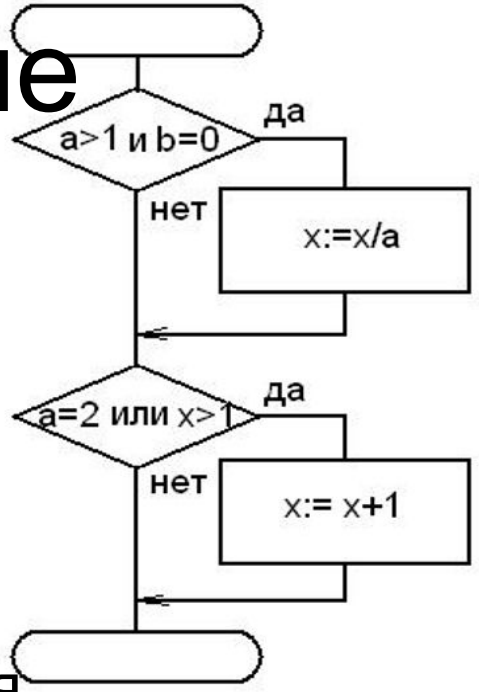
a=1, b=1, x=1 – путь 1-2-4-6, условия: 1-нет, 2-нет, 3-нет, 4-нет.

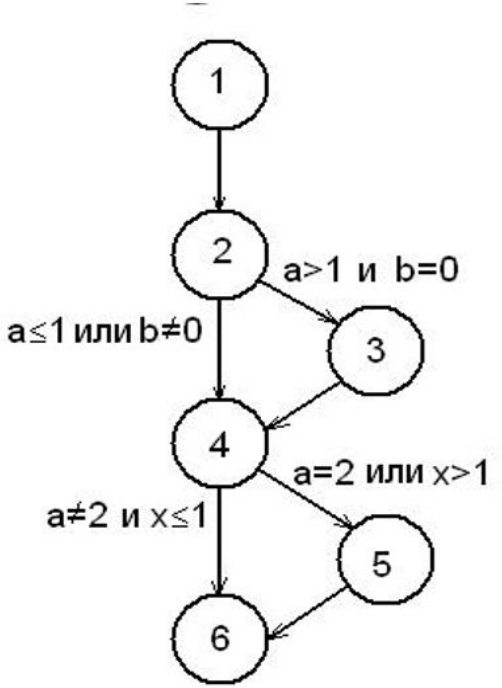




Акт

Комбинаторное покрытие





Комбинаторное покрытие условий требует покрыть тестами восемь комбинаций:

1) a>1, b = 0;
2) a>1, b ≠ 0;
3) a<1, b = 0;
4) a<1, b ≠ 0;
5) a = 2, x>1;
6) a = 2, x<1;
7) a ≠ 2, x>1;
8) a ≠ 2, x<1.

Эти комбинации можно проверить четырьмя тестами:

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).

Таким образом

- Для программ, содержащих только одно условие на каждое решение, минимальным является набор тестов, который проверяет все результаты каждого решения и передает управление каждому оператору, по крайней мере, один раз.
- Для программ, содержащих вычисления, каждое из которых требует проверки более чем одного условия, минимальный набор тестов должен:
  - ✔ генерировать все возможные комбинации результатов проверок условий для каждого вычисления;
  - ✔ передавать управление каждому оператору, по крайней мере, один раз.

Тестовое покрытие циклов

- При проверке циклов основное внимание обращается на правильность конструкций циклов.
- Количество наборов тестов для проверки циклов по принципу «белого ящика» зависит от типа цикла.
- Различают 4 типа циклов:
  1) простые,
  2) вложенные,
  3) объединенные,
  4) неструктурированные.

Тестовое покрытие циклов

- Для проверки простых циклов с количеством повторений n может использоваться один из следующих наборов тестов:

1) прогон всего цикла;

2) только один проход цикла;

3) два прохода цикла;

4) t проходов цикла, где $t < n$;

5) $n - 1$, n, $n + 1$ проходов цикла.

Тестовое покрытие циклов

- С увеличением уровня вложенности циклов количество возможных путей резко возрастает. Это приводит к нереализуемому количеству тестов. Для сокращения количества тестов применяется методика, использующа понятия объемлющего и вложенного циклов.

Порядок тестирования вложенных циклов следующий:

- Выбирается самый внутренний цикл. Устанавливаются минимальные значения параметров всех остальных циклов.
- Для внутреннего цикла проводятся тесты простого цикла. Добавляются тесты для исключенных значений и значений, выходящих за пределы рабочего диапазона.
- Переходят в следующий по порядку объемлющий цикл. Выполняют его тестирование. При этом сохраняются минимальные значения параметров для всех внешних (объемлющих) циклов и типовые значения для всех вложенных циклов.
- Работа продолжается до тех пор, пока не будут протестированы все циклы.