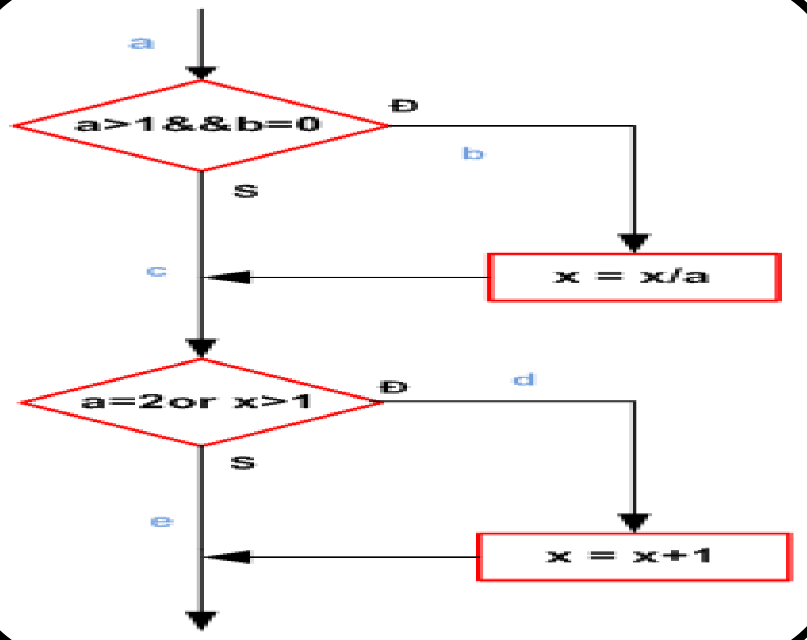
# Code Coverage

- **Function coverage** - Has each function (or subroutine) in the program been called?
- **Statement coverage** - Has each node in the program been executed?
- **Decision coverage** - Has every edge in the program been executed? For instance, have the requirements of each branch of each control structure (such as in IF and CASE statements) been met as well as not met?
- **Path coverage –** Has every path in the program been executed ?
- **Condition coverage** (or predicate coverage) - Has each Boolean sub-expression evaluated both to true and false? This does not necessarily imply decision coverage.
- **Condition/decision coverage** - Both decision and condition coverage should be satisfied.
- **Modified condition/decision coverage**

# Code Coverage(tt)

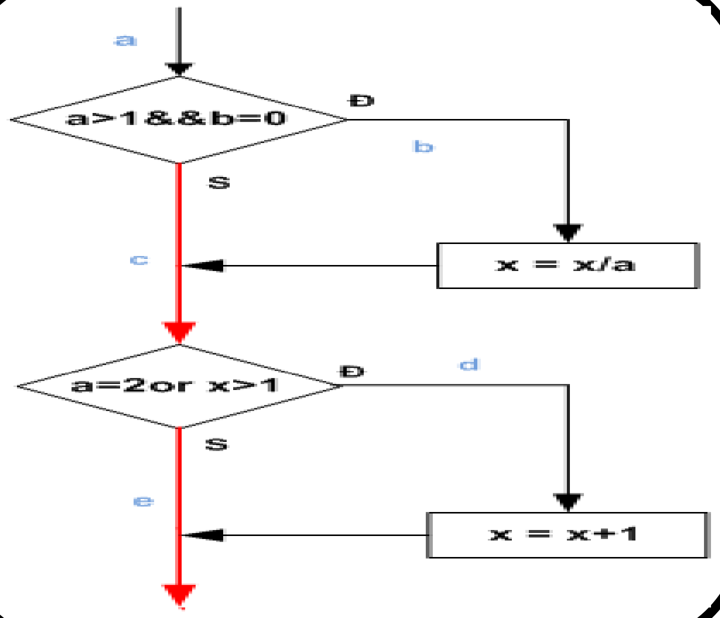
**1. Statement coverage** - Has each node in the program been executed?

```
void foo(int a, int b, int x)
{
    if ( (a > 1) &  &   (b ==  0 ) )
          x = x / a;
    if(a == 2  ||  x>1)
          x = x + 1;
}
```



# Code Coverage(tt)

2. **Decision coverage** - Has every edge in the program been executed? For instance, have the requirements of each branch of each control structure (such as in IF and CASE statements) been met as well as not met?



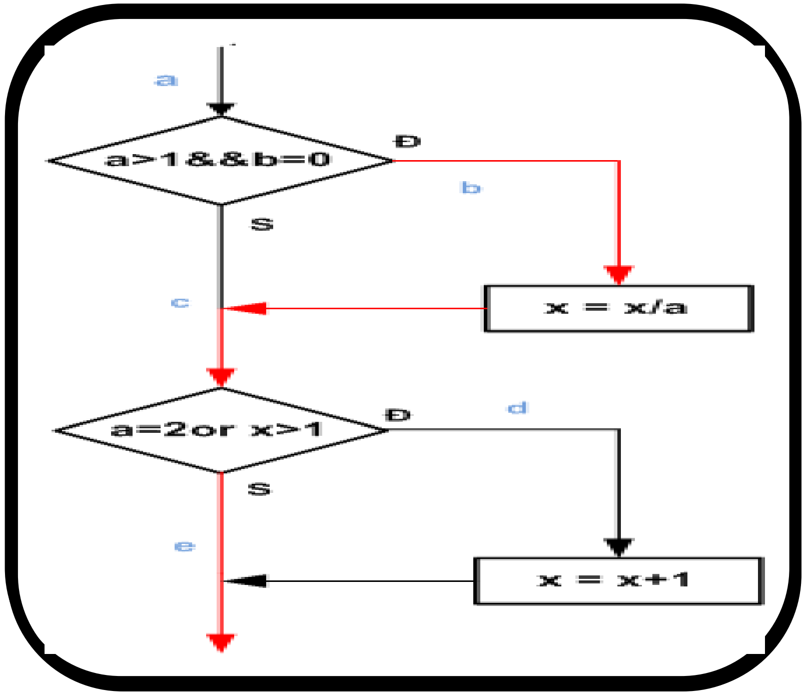
1. a = 2, b = 0, x = 4
2. a = 0, b = 0, x = 0

➡ Decision coverage ?

Decision coverage → Statement coverage ?

# Code Coverage(tt)

3. **Path coverage –** Has every path in the program been executed ?



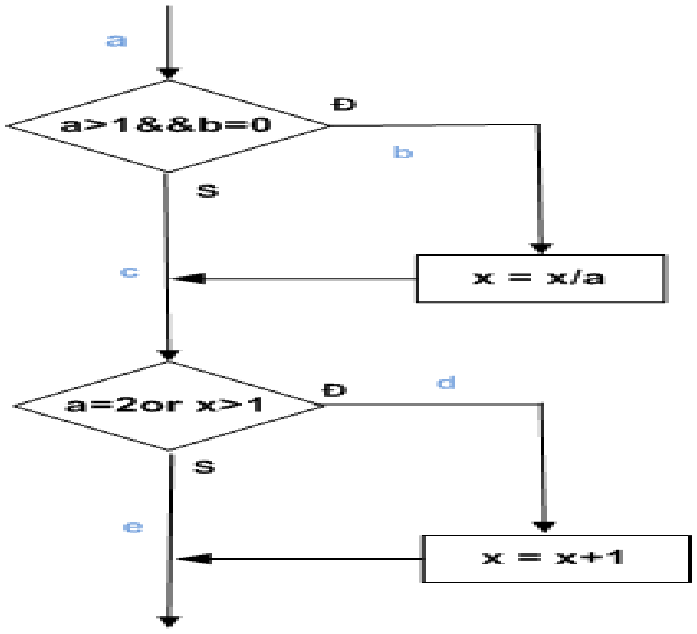
1. a = 2, b = 0, x = 4
2. a = 0, b = 0, x = 0
3. a = 2, b = 1, x = 4
4. a = 3, b = 0, x = 0

$\Rightarrow$ Path coverage ?

Path coverage → Decision coverage ?

# Code Coverage(tt)

**4. Condition coverage** (or predicate coverage) - Has each Boolean sub-expression evaluated both to true and false? This does not necessarily imply decision coverage.



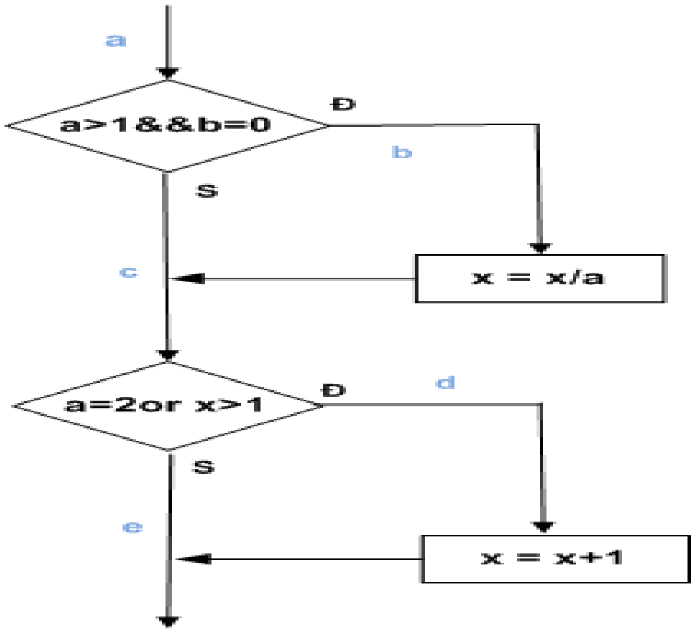
1. a = 2, b = 0, x = 4
2. a = 1, b = 1, x = 1

→ Condition coverage

Condition coverage → Path, Decision ???

# Code Coverage(tt)

**5. Condition/decision coverage** - Both decision and condition coverage should be satisfied.



1. a = 2, b = 0, x = 4
2. a = 1, b = 1, x = 1

⇨ D/C coverage

# Code Coverage(tt)

6. **Modified condition/decision**:

   For safety-critical applications (e.g., for avionics software) it is often required **that modified condition/decision coverage (MC/DC)** is satisfied. This criteria extends condition/decision criteria **with requirements that each condition should affect the decision outcome** independently. For example, consider the following code:

   **if (a or b) and c then**

*We have two test cases:*

- A = true , b = true , c = true
- A = false, b = false, c = false

D/C coverage but not MC/DC coverage ?
However, the above tests set will not satisfy modified condition/decision coverage, since in the first test, the value of 'b' and in the second test the value of 'c' would not influence the output

# Code Coverage(tt)

- **if (a or b) and c then**
- So, the following test set is needed to satisfy MC/DC:
  - a=**false**, b=**false**, c=true
  - a=**true**, b=false, c=**true**
  - a=false, b=**true**, c=**true**
  - a=true, b=true, c=**false**
- The bold values influence the output, each variable must be present as an influencing value at least once with false and once with true.

# Code Coverage(tt)

- **if (a or b) and c then**
- So, the following test set is needed to satisfy MC/DC:
  - a=**false**, b=**false**, c=true
  - a=**true**, b=false, c=**true**
  - a=false, b=**true**, c=**true**
  - a=true, b=true, c=**false**
- The bold values influence the output, each variable must be present as an influencing value at least once with false and once with true.

# Code Coverage(tt)

- **Multiple condition coverage**

  **Code: if (a or b) and c then**

- This criteria requires that all combinations of conditions inside each decision are tested. For example, the code fragment from the previous section will require eight tests:
  - a=false, b=false, c=false
  - a=false, b=false, c=true
  - a=false, b=true, c=false
  - a=false, b=true, c=true
  - a=true, b=false, c=false
  - a=true, b=false, c=true
  - a=true, b=true, c=false
  - a=true, b=true, c=true

## Exercise 1

```
• if (A<B) and (A>2) then
      X        A
  else
      X        B
  end-if
   if (B>0) and (A>0) then
      Y        A + B
  else
      Y        0
  end-if
```

# Exercise 2

-
  ```
  int foo(int a, int b)
  {     int c = b;
        if ((a>5) && (b>0))
                  { c = a; }
        return a*c;
  }
  ```

## Exercise 3

```
public void method( int x, int count ) {
1       while ( x > 10 )
2           x -= 10;
3       if ( count < 20 && x%2 == 0 )
4           count += 20;
        else
5           count -= 20;
    }
```

## Exercise 4

- ( ((u == 0) || (x>5)) && ((y<6) || (z == 0))

| Test case n° | A: (u == 0) | B: (x>5) | C: (y<6) | D: (z == 0) | ( (A \|\| B) && (C \|\| D) ) |
|---|---|---|---|---|---|
| 1 | F | F | F | F | F |
| 2 | F | F | F | T | F |
| 3 | F | F | T | _ | F |
| 4 | F | T | F | F | F |
| 5 | F | T | F | T | T |
| 6 | F | T | T | _ | T |
| 7 | T | _ | F | F | F |
| 8 | T | _ | F | T | T |
| 9 | T | _ | T | _ | T |