* **Branch coverage is also known as Decision coverage or all-edges coverage.**
* **It covers both the true and false conditions unlikely the statement coverage**

**Advantages of decision coverage:**

* **To validate that all the branches in the code are reached**
* **To ensure that no branches lead to any abnormality of the program’s operation**

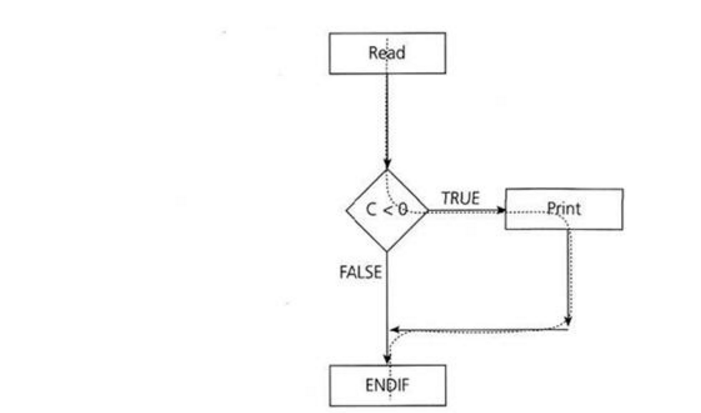
**In the previous section we saw that just one test case was required to achieve 100% statement coverage. However, decision coverage requires each decision to have had both a True and False outcome. Therefore, to achieve 100% decision coverage, a second test case is necessary where A is less than or equal to B which ensures that the decision statement ‘IF A > B’ has a False outcome. So one test is sufficient for 100% statement coverage, but two tests are needed for 100% decision coverage. It is really very important to note that 100% decision coverage guarantees 100% statement coverage, but notthe other way around.**

**1 READ A  
2 READ B  
3 C = A – 2 \*B  
4 IF C <0 THEN  
5 PRINT “C negative”  
6 ENDIF**

**Let’s suppose that we already have the following test, which gives us 100% statement coverage for code sample 4.3.**

**TEST SET 2   Test 2\_1: A = 20, B = 15 C = -10**

**The value of C is -10, so the condition  ‘C < 0’ is True, so we will print ‘C negative’ and we have executed the True outcome from that decision statement. But we have not executed the False outcome of the decision statement. What other test would we need to exercise the False outcome and to achieve 100% decision coverage?**

****

**The dotted line shows where Test 2\_1 has gone and clearly shows that we haven’t yet had a test that takes the False exit from the IF statement.  
Let’s modify our existing test set by adding another test.**

**TEST SET 2  
Test 2\_1: A = 20, B = 15 C = -10  
Test 2\_2: A = 10, B = 2 C = 6**

**This now covers both of the decision outcomes, True (with Test 2\_1) and False (with Test 2\_2). If we were to draw the path taken by Test 2\_2, it would be a straight line from the read statement down the False exit and through the ENDIF. We could also have chosen other numbers to achieve either the True or False outcomes.**