Control Flow Testing

Structural Testing

- In structural testing, the software is viewed as a white box and test cases are determined from the implementation of the software.
- Structural testing techniques include control flow testing and data flow testing.

Control Flow Testing

- Control flow testing uses the control structure of a program to develop the test cases for the program.
- The test cases are developed to sufficiently cover the whole control structure of the program.
- The control structure of a program can be represented by the control flow graph of the program.

Control Flow Graph

- The control flow graph G = (N, E) of a program consists of a set of nodes N and a set of edge E.
- Each node represents a set of program statements. There are five types of nodes. There is a unique entry node and a unique exit node.
- There is an edge from node n_1 to node n_2 if the control may flow from the last statement in n_1 to the first statement in n_2.

Control Flow Graph: Nodes

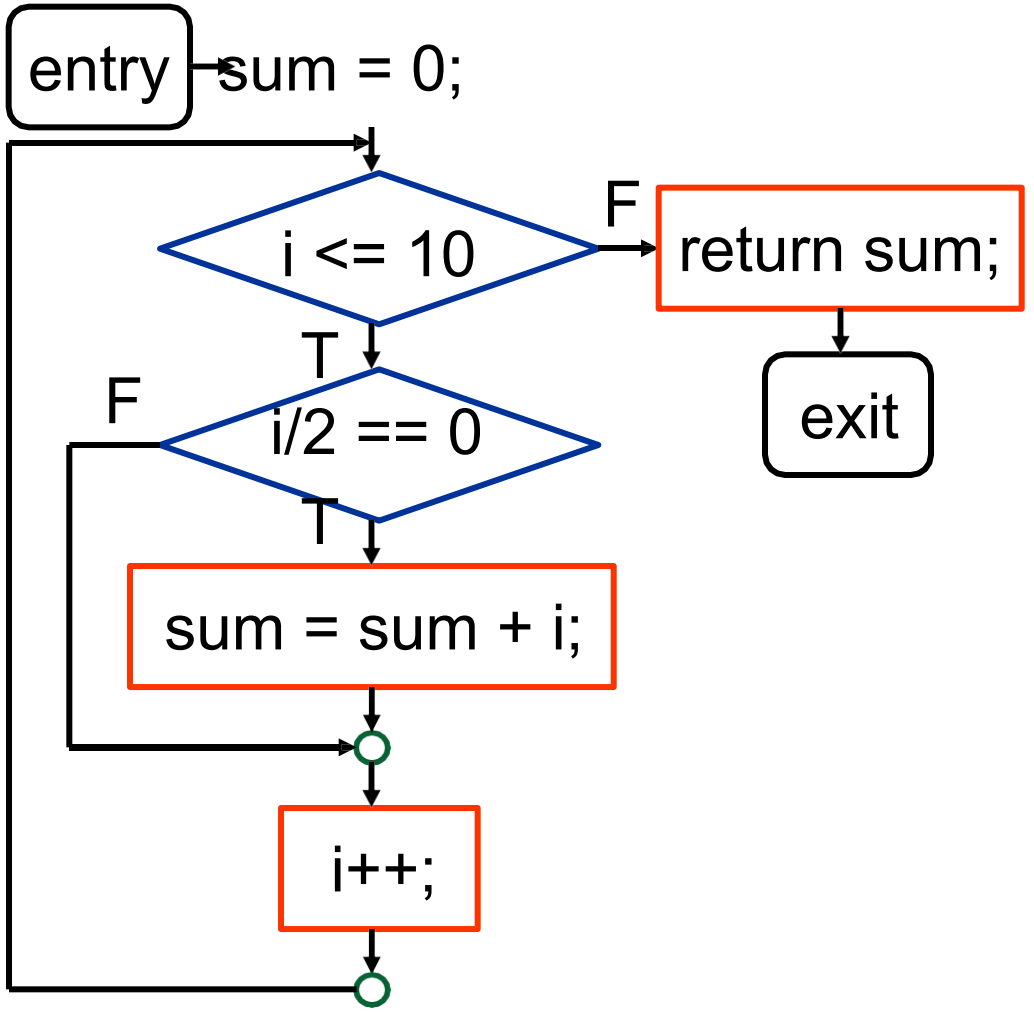
- A decision node contains a conditional statement that creates 2 or more control branches (e.g. if or switch statements).
- A merge node usually does not contain any statement and is used to represent a program point where multiple control branches merge.
- A statement node contains a sequence of statements. The control must enter from the first statement and exit from the last statement.

Control Flow Graph: An Example

```
int evensum(int i)
{
  int sum = 0;

  while (i <= 10) {
    if (i/2 == 0)
      sum = sum + i;
    i++;
  }
  return sum;

}
```



Test Cases

- A test case is a complete path from the entry node to the exit node of a control flow graph.
- A test coverage criterion measures the extent to which a set of test cases covers a program.

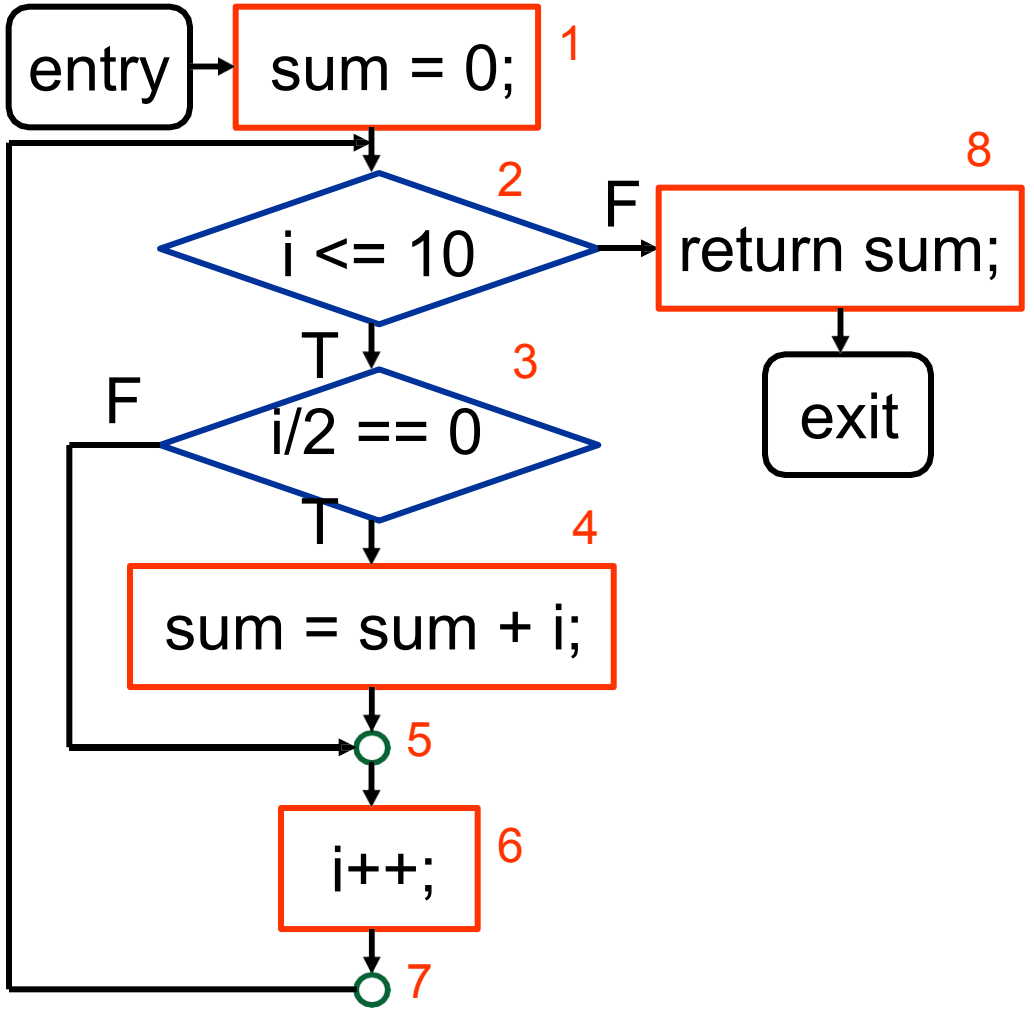
Test Coverage Criteria

- Statement coverage (SC) — Node coverage
- Decision coverage (DC) — Edge coverage
- Condition coverage (CC)
- Decision/condition coverage (D/CC)
- Multiple condition coverage (MCC)
- Path coverage (PC)

Statement Coverage

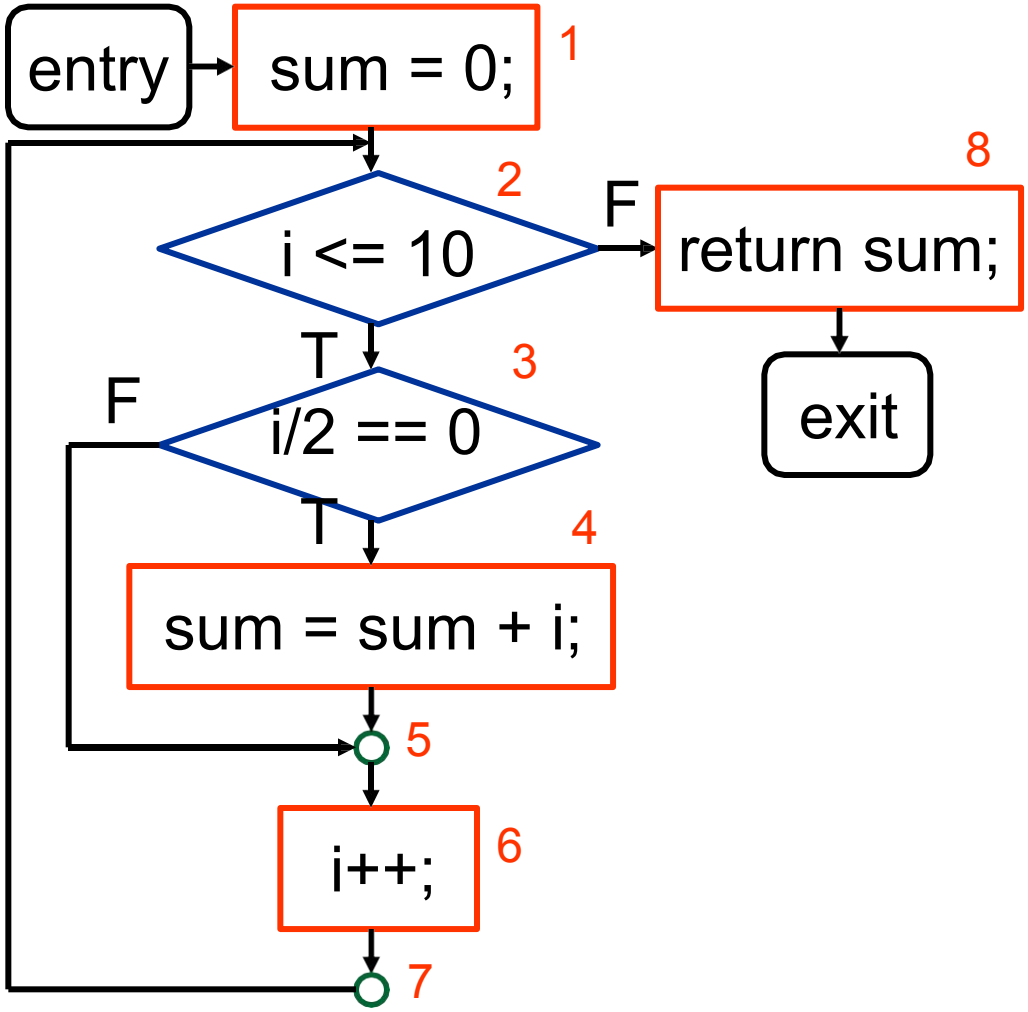
- Every statement in the program has been executed at least once.

1 → 2 → 3 → 4 →
5 → 6 → 7 → 2 → 8



Decision Coverage

- Every statement in the program has been executed at least once, and every decision in the program has taken all possible outcomes at least once.



1 → 2 → 3 → 5 → 6 → 7 → 2 →
3 → 4 → 5 → 6 → 7 → 2 → 8

Condition Coverage

- Every statement in the program has been executed at least once, and every condition in each decision has taken all possible outcomes at least once.

Decision/Condition Coverage

- Every statement in the program has been executed at least once, every decision in the program has taken all possible outcomes at least once, and every condition in each decision has taken all possible outcomes at least once.

Multiple Condition Coverage

- Every statement in the program has been executed at least once, all possible combination of condition outcomes in each decision has been invoked at least once.

Example

To demonstrate the different white-box logic coverage techniques of statement, decision, and condition coverage, the piece of code shown in Figure 1 will be used.

```
A = 300
if B > 40 and C < 100 then A = 1000
if B < 60 and C < 20 then A = 10
print A
```

Figure 1: Code with the Input Variables B and C

Statement Coverage:

To have complete statement coverage, each statement must be executed at least once.

Test Case #	Inputs		Expected Output
	B	C	A
1	> 40 and < 60	<20	10

Table 1: Statement Coverage - Example

Decision Coverage:

To have decision coverage, also called branch coverage, each statement is executed at least once and each decision takes all possible outcomes at least once.

Test Case #	Inputs		Expected Output
	B	C	A
2	>= 60	<20	1000
3	<= 40	<20	10

Table 2: Decision Coverage - Example

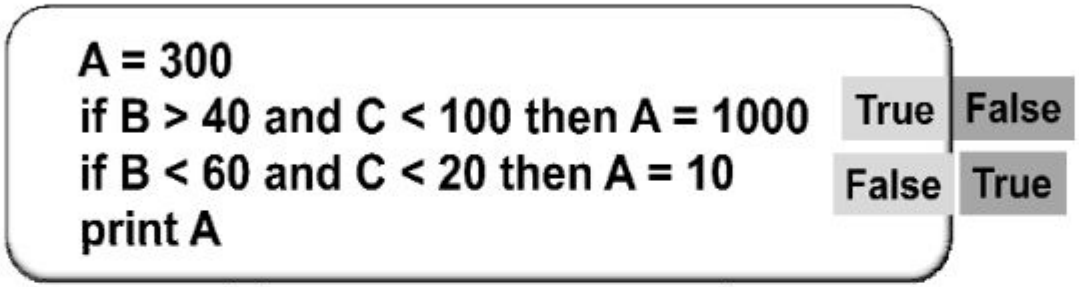


Figure 2: Decision Coverage Test Case Results – Test Cases 2 & 3

Condition Coverage:

To have condition coverage, each statement is executed at least once and each condition in a decision takes all possible outcomes at least once.

For the code in Figure 1 there are three conditions that the input variable B can have:
1.$B \leq 40$
2.$40 < B < 60$
3.$B \geq 60$.

There are also three conditions that input variable C can have:
1.$C < 20$
2.$20 \leq C < 100$
3.$C \geq 100$.

Condition Coverage:

Table 3 illustrates one choice of test cases that combines these into condition coverage of the code in Figure 1.

Note that condition coverage does not always imply decision coverage.

Test Case #	Inputs		Expected Output
	B	C	A
4	>= 60	<20	1000
5	<= 40	>= 20 and <100	300
6	> 40 and < 60	>= 100	300

Table 3: Condition Coverage - Example

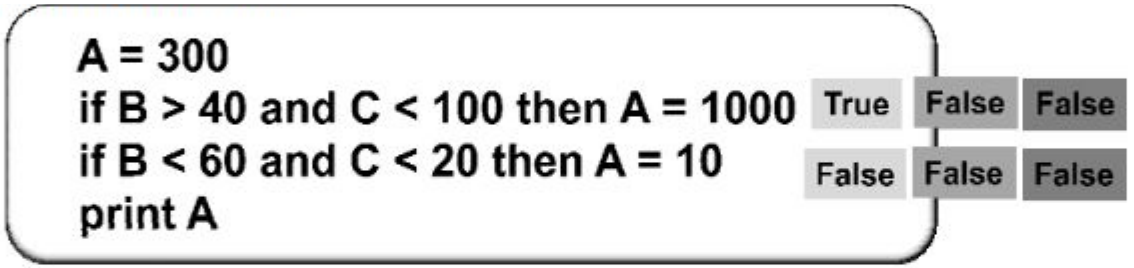


Figure 3: Condition Coverage Test Case Results – Test Cases 4, 5 & 6

Condition and Decision Coverage:

The next level of rigor is to have condition and decision coverage where each statement is executed at least once, each decision takes all possible outcomes at least once, and each condition in a decision takes all possible outcomes at least once.

Test Case #	Inputs		Expected Output
	B	C	A
7	>= 60	>=20 and < 100	1000
8	> 40 and < 60	<20	10
9	<= 40	>= 100	300

Table 4: Condition & Decision Coverage - Example

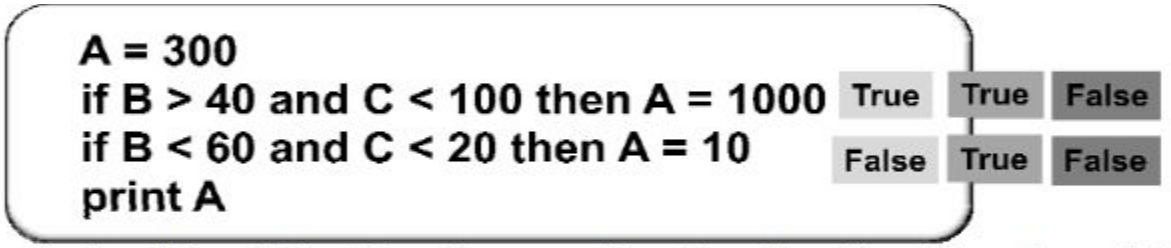


Figure 4: Condition & Decision Coverage Test Case Results – Test Cases 7, 8 & 9

Multiple Condition Coverage:

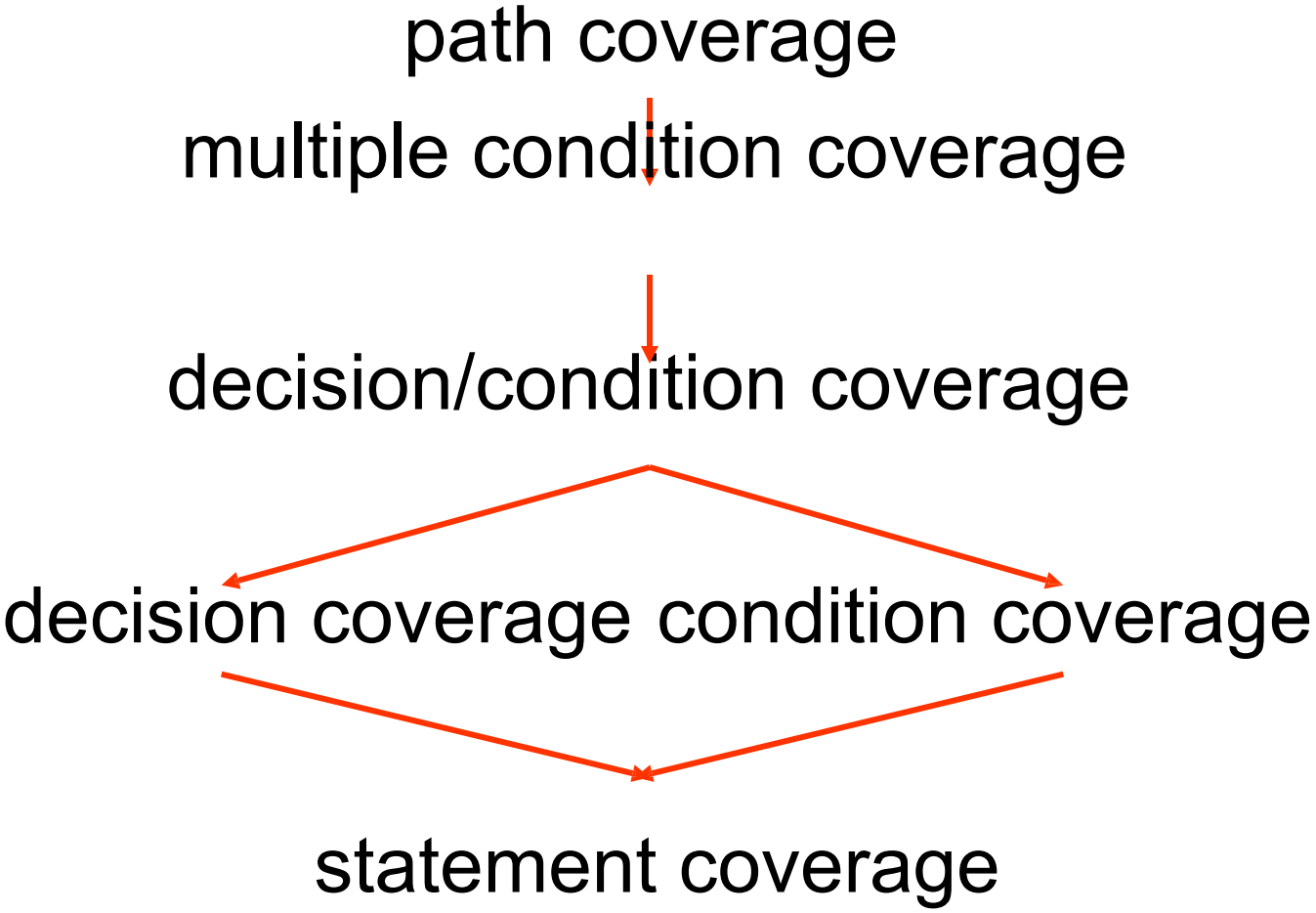
To have multiple condition coverage, each statement is executed at least once and all possible combinations of condition outcomes in each decision occur at least once. Multiple condition coverage always results in condition, decision, and statement coverage as well. Multiple condition coverage is the most rigorous type of structural coverage testing.

Test Case #	Inputs		Expected Output
	B	C	A
1	<= 40	<20	10
2	<= 40	>= 20 and <100	300
3	<= 40	>= 100	300
4	> 40 and <60	<20	10
5	> 40 and <60	>= 20 and <100	1000
6	> 40 and <60	>= 100	300
7	>= 60	<20	1000
8	>= 60	>= 20 and <100	1000
9	>= 60	>= 100	300

Path Coverage

- Every complete path in the program has been executed at least once.
- A loop usually has an infinite number of complete paths.

Test Coverage Criteria Hierarchy



Testing Simple Loops

- Skip the loop entirely
- Go once through the loop
- Go twice through the loop
- If the loop has max passes = n, then go n – 1, n, and n + 1 times through the loop

Testing Nested Loops

- Set all outer loops to their minimal value and test the innermost loop
- Add tests of out-of-range values
- Work outward, at each stage holding all outer loops at their minimal value
- Continue until all loops are tested

Java Code Coverage Tool

- EclEmma is a free Java code coverage tool for Eclipse

  http://www.eclemma.org

- EclEmma adopts the philosophy of the EMMA Java code coverage tool for the Eclipse workbench

  http://emme.sourceforge.net

EclEmma

- Fast develop/test cycle: Launches from within the workbench like JUnit and test runs can directly be analyzed for code coverage.
- Rich coverage analysis: Coverage results are immediately summarized and highlighted in the Java source code editors.
- Non-invasive: EclEmma does not require modifying your projects or performing any other setup.