Code Coverage

Introduction

- Code coverage measures the number of lines of source code executed during a given test suite for a program.
- Tools that measure code coverage normally express this metric as a percentage.
- Code Coverage utilities hook into your source code and your test suite and return statistics on how much of your code is actually covered by your tests.

Types of code coverages

The common metrics that you might see mentioned in your coverage reports include:

- Statement coverage
- Decision or Branches coverage
- Condition coverage
- Function coverage
- Function call coverage
- Line Coverage
- Loop Coverage

Statement coverage

- This is a metric that ensures that each statement of the code is executed at least once.
- It measures the number of lines executed.
- What is the use of statement coverage?
  - We can find the dead codes
  - We can find the unused branches

Example of statement coverage

- In first test case , else part is being executed i.e., 8 statements out of 11 is being executed
- Similarly , if part is being executed means 7 out of 11 statements is being executed
- If we combine both cases , we will cover 100% statements

```
void test_func( bool condition )
{
 if ( condition == true )
{
printf("Condition is true\n");
}
else
 {
   printf("Condition is false\n");
 }
}
```

```
void test_func( bool condition )
{
 if ( condition == true )
 {
   printf("Condition is true\n");
 }
else
{
printf("Condition is false\n");
}
}
```

Decision or branch coverage

- Branch coverage ensures each branch in the program (e.g., if statements, loops) has been executed.
- Each branch must be executed at least once during testing
- By taking the example of statement coverage , if both th “if” and “else” is covered we can say we have 100% branch coverage

Difference between statement and branch coverage

- When the condition is true, we have 100% statement coverage
- But we the condition is if the condition is false we may not have the branch coverage
- If we achieve 100% of branch coverage, that means we have covered all the statements too.
- But if we achieve 100% of statement coverage, that doesn't mean, we have covered all the branches as well.

```
void test_func( bool condition )
{
 if ( condition == true )
 {
    printf("Condition is true\n");
 }
 printf("EmbeTronicX\n");
}
```

Condition coverage

- Condition coverage only applies to logical operands like AND, OR, XOR.
- In order to ensure complete Condition coverage criteria, the logical operands should be evaluated at least once against “true” and “false“.
- **Example : if(X && Y)**
  - To get a valid condition coverage we may check following tests
    - TEST 1: X=TRUE, Y=FALSE
    - TEST 2: X=FALSE, Y=TRUE

Function and function call coverage

- **Function Coverage :**
  - It refers to the number of functions in your code that were tested.

- **Function call coverage :**
  - It is a very common scenario in programming that one function calls another and so on. There is a calling function and a called function. So this coverage technique ensures that there do not exist any faults in the function call.

Line and loop coverage

- Line Coverage is straightforward. It’s the number of lines of code your tests evaluated.
- Loop coverage :
  - This technique is used to ensure that all the loops have been executed, and the number of times they have been executed.
  - The purpose of this coverage technique is to make sure that the loops don’t iterate infinitely or terminate abnormally.
  - Loop testing aims at monitoring the beginning until the end of the loop.