Structural Testing

Structural Testing is a white-box testing technique that focuses on the internal structure and logic of the code. It ensures that all statements, branches, and paths in the program are executed at least once.

Purpose of Functional Testing

- Focuses on the internal code structure rather than external behavior. Validates that all functional requirements are met.
- Ensures that all possible execution paths are tested.
- Ensures decision points, loops & branches work correctly
- Used to detect logical errors, unreachable code, and inefficiencies.

Control Flow Graph (CFG) Representation

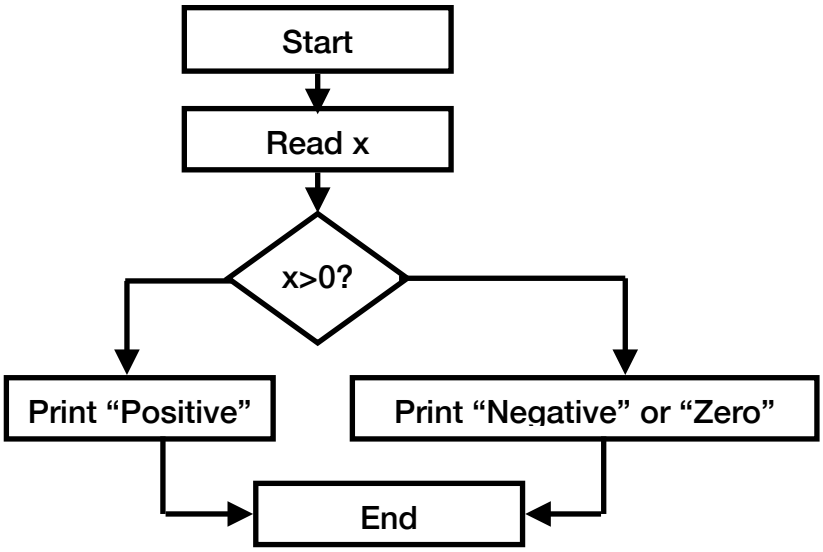
A Control Flow Graph (CFG) is created from the program code. It consists of:

- Nodes – Represent code statements or decision points.
- Edges – Represent the flow of execution.
- Entry & Exit Points – Indicate the start and end of the program.

Example: Simple If-Else Statement

```
void checkNumber(int x) {
    if (x > 0) {
        printf("Positive");
    } else {
        printf("Negative or Zero");
    }
}
```

Test Cases for 100% Coverage



Test Case	Input (x)	Expected Output
Case 1	5	"Positive"
Case 2	-3	"Negative or Zero"

1. Statement Coverage

Ensures every line of code is executed at least once.

$$StatementCoverage = \frac{Numberofexecutedstatements}{Totalstatements} * 100$$

Example: If a program has 10 statements and we executed 8 of them

$$StatementCoverage = \frac{8}{10} * 100 = 80\,\%$$

```
void checkNumber(int x) {
    printf("Start\n");   // Statement 1
    if (x > 0) {         // Statement 2
        printf("Positive\n"); // Statement 3
    }
    printf("End\n");     // Statement 4
}
```

Test Case	Input (x)	Executed Statements
Case 1	5	1,2,3,4
Case 2	-3	1,2,4

2. Branch Coverage (Decision Coverage)

Ensures that every decision (if, else, switch, loop) has been executed for both True and False outcomes.

Example:

```
void checkEvenOdd(int num) {
    if (num % 2 == 0) {
        printf("Even\n");  // Branch 1
    } else {
        printf("Odd\n");   // Branch 2
    }
}
```

Test Case	Input (num)	Executed Branch
Case 1	4	Even (Branch 1)
Case 2	7	Odd (Branch 2)

3. Condition Coverage

Ensures that each Boolean condition in a decision statement evaluates to both True and False at least once.

Example:

```
void checkValues(int a, int b) {
    if (a > 0 && b < 10) {
        printf("Valid Input\n");
    }
}
```

Test Case	Input (a,b)	Condition Evalutions
Case 1	5, 8	True (a>0), True (b<10)
Case 2	-2, 8	False (a>0), True (b<10)

Test Case	Input (a,b)	Condition Evalutions
Case 3	5, 12	True (a>0), False (b<10)

4. Path Coverage

Ensures all possible execution paths are tested. More comprehensive than statement and branch coverage.
Example:

```
void processNumber(int x) {
    if (x > 10) {
        printf("Greater than 10\n");
    } else if (x == 10) {
        printf("Equal to 10\n");
    } else {
        printf("Less than 10\n");
    }
}
```

Test Case	Input (x)	Path Taken
Case 1	15	Path 1
Case 2	10	Path 2
Case 3	5	Path 3

5. Loop Testing

Ensures correct loop execution for:

1. Zero iterations (loop doesn't execute).
2. One iteration.
3. Multiple iterations.

Example:

```
void printNumbers(int n) {
    for (int i = 0; i < n; i++) {
        printf("%d", i);
    }
}
```

Test Case	Input (x)	Expected Output
Case 1	0	No output (Zero Iterations)
Case 2	1	"0" (One Iteration)
Case 3	3	"012" (Multiple Iterations)