For Loop Statements

- Loop statements allow you to execute a statement, or a group of statements, multiple times
- Types of loops in C programming:
  - **for** loop
  - **while** loop
  - **do...while** loop
- Jump statements alter the normal execution sequence of a program:
  - **continue** - skip some statements inside the loop
  - **break** - terminate the execution of the remaining loop statements
  - **goto** - jump from one statement to another within a function

Syntax

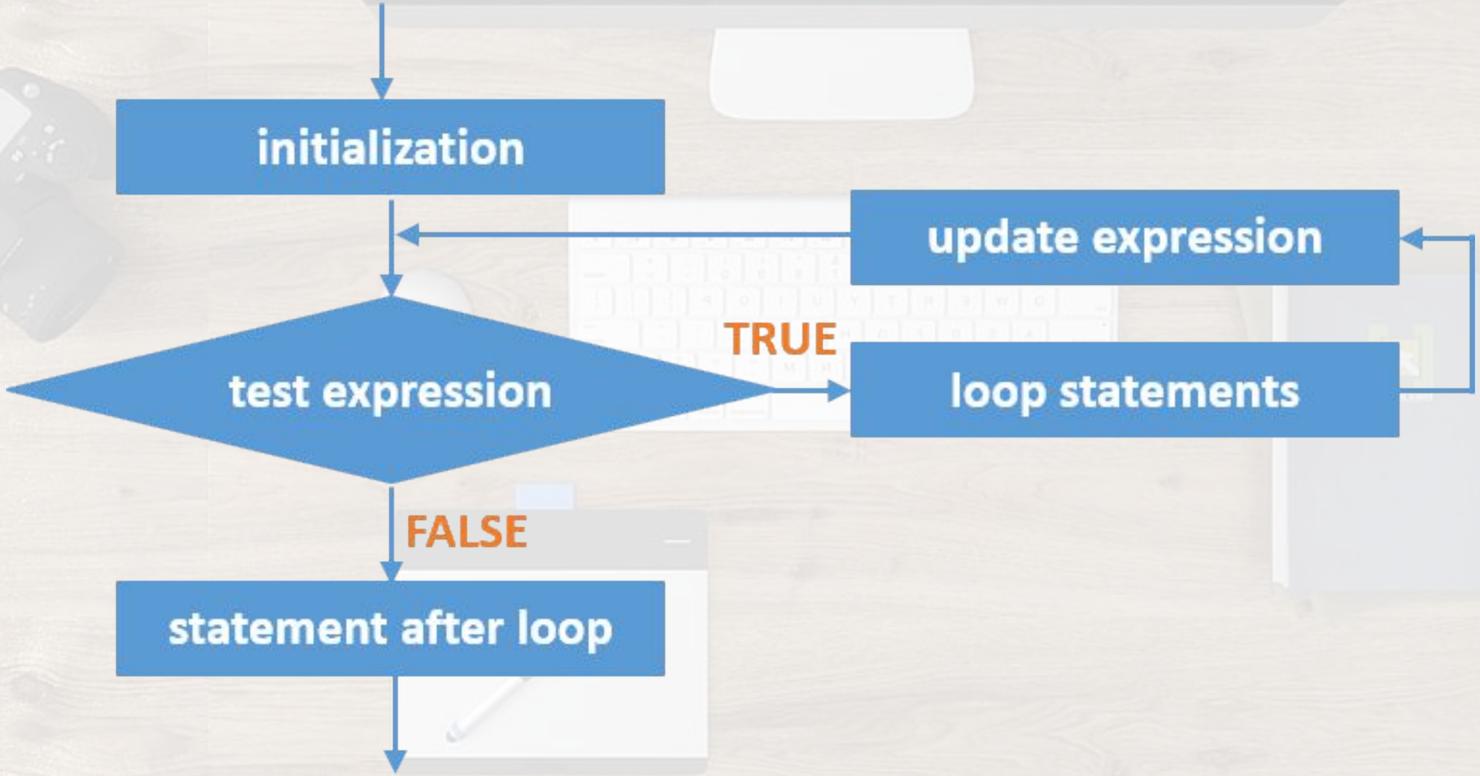
- **Syntax**:

```
for (int i = 1;i <= 5;i++) {
    printf(“i = %d\n”, i);
}
```

- **Output**:

```
i = 1
i = 2
i = 3
i = 4
i = 5
```

Flowchart


initialization
update expression
TRUE
test expression
loop statements
FALSE
statement after loop

Order of operations

1. The **initialization** is executed first and only once to initialize the loop control variable/counter
2. The **test expression** is evaluated
   a. If test expression is found to be false (0), the loop statements will not be executed. Program control jumps to the next statement after the "for" loop.
   b. If test expression is found to be true, the statements within the loop will be executed
   c. The **update expression** is executed. It will update the loop control variable/counter.
3. Step 2 is repeated until the test expression becomes false or loop is terminated using break statement.

Rules

- **for** is a keyword and must be used only in lower case letters
- **for** statement can be an empty statement
- Every **for** statement must include initialization, test expression and update expression. They can be empty but must be separated with semicolon (;)
- Example:

```
for(i=1; i<10; i++)
 {
 }
```

Result: Variable **i** is incremented