In **C programming**, a **statement** is a complete instruction that tells the compiler to perform a specific action.
Statements are executed **sequentially** unless control flow changes (e.g., loops, conditionals, jumps).

Types of Statements in C

1. Expression Statements

- Perform computations or assignments.
- End with a semicolon `;`.

```
x = y + 5;
printf("Hello World\n");
```

2. Compound Statements (Block)

- Group multiple statements inside `{ }`.
- Treated as a single statement.

```
{
    int a = 5;
    int b = 10;
    printf("%d\n", a + b);
}
```

3. Control Flow Statements

Control the order in which statements execute.

a) Conditional Statements

- `if`, `if-else`, `else if`, `switch`

```
if (x > 0) {
    printf("Positive\n");
```

```
} else {
    printf("Non-positive\n");
}
```

b) Looping Statements

- for, while, do-while

```
for (int i = 0; i < 5; i++) {
    printf("%d\n", i);
}
```

c) Jump Statements

- break – Exit loop/switch.
- continue – Skip to next loop iteration.
- goto – Jump to a labeled statement (use with caution).
- return – Exit from a function.

```
for (int i = 0; i < 5; i++) {
    if (i == 3) break;
    printf("%d\n", i);
}
```

4. Declaration Statements

- Declare variables or constants.

```
int age;
const float PI = 3.14;
```

5. Null Statement

- A single semicolon ; that does nothing.

```
;
```

✅ **Key Points:**

- Every statement **must** end with a semicolon (except compound blocks and control structures themselves).
- Statements are executed **top to bottom** unless altered by control flow.
- Grouping statements in { } is essential when multiple statements follow a control structure.

If you want, I can prepare a **C program that demonstrates all types of statements** in one example so you can run and see them in action.
Do you want me to create that?