C PROGRAMMING HANDBOOK BY SAICHANDAN GORLI

CHAPTER 4 : LOOP CONTROL INSTRUCTION

QUICK LINKS

-

WHY LOOP

Sometimes we want our programs to execute few sets of instructions over and over again. For example: Print 1 - 100, first 100 even numbers etc.

Hence loops make it easy for a programmer to tell computer that a given set of intructions must be executed repeatedly.

TYPE OF LOOPS

Primarily there are three types of loops in C language :

1. for loop
2. while loop
3. do-while loop

FOR LOOP

```
for (initialization; condition; increment) {
    // Code to execute while condition is true
}
```

Components:

1. **initialization:** This is executed once at the beginning of the loop. It's typically used to initialize a loop counter.
2. **condition:** The loop continues to execute as long as this condition evaluates to true.
3. **increment:** This is executed at the end of each loop iteration. It's usually used to update the loop counter.

Example :

```
#include <stdio.h>

int main() {
    for (int i = 1; i <= 5; i++) {
        printf("Count: %d\n", i);
    }

    return 0;
}
```

WHILE LOOP

```
while (condition) {
    // Code to execute while condition is true
}
```

Key Points: *condition:* This is a boolean expression. The loop continues as long as this condition evaluates to true. If the condition is false from the beginning, the code inside the loop will not execute at all.

Example :

```c
#include <stdio.h>

int main() {
    int count = 1;

    while (count <= 5) {
        printf("Count: %d\n", count);
        count++; // Increment count to avoid infinite loop
    }

    return 0;
}
```

DO-WHILE LOOP

```c
do {
    // Code to execute
} while (condition);
```

Key Points:

- The code inside the do block is executed once before the condition is checked.
- If the condition evaluates to true, the loop continues; otherwise, it exits.
- The condition is checked after the loop body is executed.

Example :

```c
#include <stdio.h>

#include <stdio.h>

int main() {
    int count = 1;

    do {
        printf("Count: %d\n", count);
        count++; // Increment count
    } while (count <= 5);

    return 0;
}
```

note :

1. **Initialization:**
    - Make sure your loop control variable is properly initialized before the loop starts.
2. **Condition:**
    - Ensure the loop condition will eventually become false to avoid infinite loops.
    - Double-check the logic of the condition to prevent off-by-one errors.
3. **Increment/Decrement:**
    - Always update the loop control variable within the loop to prevent infinite loops.
    - Ensure that the increment or decrement will lead to the termination of the loop.
4. **Infinite Loops:**
    - Be cautious of conditions that might never become false. Use debugging techniques to identify infinite loops.