Looping in c

Looping refers to the repetition of a process multiple times until a certain condition is satisfied. In the C language, there are three types of loops commonly used.

There are primarily two types of loops in C programming:

- **Entry Controlled Loops:** In these loops, the condition is checked before the loop body is executed. Both the for loop and while loop are examples of entry-controlled loops.
- **Exit Controlled Loops:** In exit-controlled loops, the condition is evaluated after the loop body has been executed. This ensures that the loop body runs at least once, regardless of whether the condition is true or false. The do-while loop is an example of an exit-controlled loop.

Types of Looping in C

There are three types of loops in C language that is given below:

- for loop
- Nesting of For loop
- while loop
- do-while loop

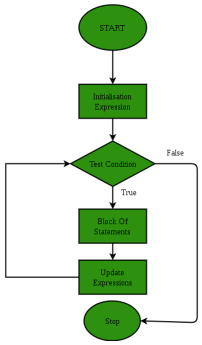
for loop

A for loop is a control structure in programming that allows for repeated execution of a block of code for a specific number of iterations. It typically consists of three parts: initialization, condition, and increment/decrement. The loop continues executing as long as the condition remains true, making it ideal for cases where the number of iterations is known beforehand.

Syntax:

```
for (initialization; condition; increment/decrement) {
    // Code to be repeated
}
```

Flowchart of for loop



Example:

```
#include <stdio.h>
```

```
int main() {
    // For loop to print numbers from 1 to 5
    for (int i = 1; i <= 5; i++) {
        printf("%d\n", i);
    }

    return 0;
}
```

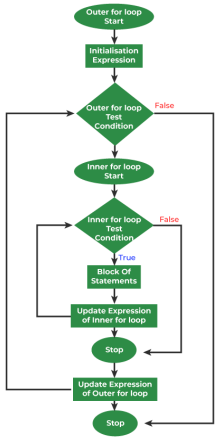
Nesting of For loop

In C programming, nested for loops occur when one loop is placed inside another loop. The inner loop runs completely for every single iteration of the outer loop, making this structure useful when dealing with multidimensional data, tables, or complex iterations.

Syntax:

```
for (initialization; condition; increment/decrement) {
    for (initialization; condition; increment/decrement) {
        // Code to be executed
    }
}
```

Flowchart of Nesting of for loop



Example:

```
#include <stdio.h>
int main() {
    // Outer loop for rows
    for (int i = 1; i <= 3; i++) {
        // Inner loop for columns
        for (int j = 1; j <= 3; j++) {
            printf("(%d, %d) ", i, j);
        }
```

```
        printf("\n");  // Move to the next line after inner loop completes
    }

    return 0;
}
```

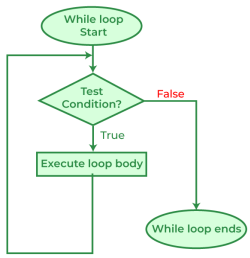
While loop

In C programming, a while loop is a control structure used to repeat a block of code as long as a specified condition remains true. Unlike the for loop, the while loop checks the condition before executing the block of code, making it ideal for situations where the number of iterations is not known beforehand.

Syntax:

```
while (condition) {
    // Code to be executed repeatedly
}
```

Flowchart of While loop



Example:

```
#include <stdio.h>
int main() {
    int i = 1;

    // While loop to print numbers from 1 to 5
    while (i <= 5) {
        printf("%d\n", i);
        i++;  // Increment the counter
    }

    return 0;
}
```

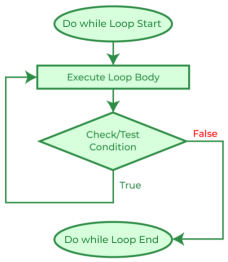
Do-while Loop

In C programming, the do-while loop is similar to the while loop, but with a key difference: the code inside the loop is executed at least once before the condition is checked. The condition is evaluated after each iteration, meaning the loop will always run at least one time, even if the condition is initially false.

Syntax:

```
do {
    // Code to be executed
} while (condition);
```

Flowchart of do while loop



Example:

```
#include <stdio.h>
int main() {
    int i = 1;

    // Do-while loop to print numbers from 1 to 5
    do {
        printf("%d\n", i);
        i++;  // Increment the counter
    } while (i <= 5);

    return 0;
}
```