1. Introduction to Looping in C

A **loop** is a flow control statement that allows a block of code to be **executed repeatedly** as long as a given condition is true.
Loops reduce code repetition and make programs efficient and readable.

Why Loops Are Needed

- To perform repetitive tasks
- To process arrays and strings
- To reduce program length
- To improve program clarity

---

2. Types of Looping Statements in C

C language provides **three types of loops**:

1. **while loop**
2. **do–while loop**
3. **for loop**

All loops consist of:

- Initialization
- Condition checking
- Execution of statements
- Increment / Decrement

---

3. while Loop

The **while loop** is an **entry-controlled loop**.
The condition is checked **before** executing the loop body.

Syntax

```
while(condition)
{
   statements;
}
```

Working of while Loop

1. Condition is checked

2. If condition is true → loop body executes
3. Condition is checked again
4. Loop continues until condition becomes false

Example

```
int i = 1;
while(i <= 5)
{
   printf("%d ", i);
   i++;
}
```

Uses

- When number of iterations is not known in advance

---

4. do–while Loop

The **do–while loop** is an **exit-controlled loop**.
The loop body executes **at least once**, even if the condition is false.

Syntax

```
do
{
   statements;
}
while(condition);
```

Example

```
int i = 1;
do
{
   printf("%d ", i);
   i++;
}
while(i <= 5);
```

Difference from while Loop

- while: condition checked first
- do–while: condition checked last

---

5. for Loop

The **for loop** is best used when the **number of iterations is known**.

Syntax

```
for(initialization; condition; increment/decrement)
{
   statements;
}
```

Example

```
int i;
for(i = 1; i <= 5; i++)
{
   printf("%d ", i);
}
```

Advantages of for Loop

- Compact and readable
- All loop control statements in one line

---

6. Comparison of Looping Statements

Feature	while	do–while	for
Condition Check	Before loop	After loop	Before loop
Minimum Execution	0 times	1 time	0 times
Use Case	Unknown iterations	At least once execution	Known iterations

---

7. Nested Loops

When one loop is placed inside another loop, it is called a **nested loop**.

Example

```
int i, j;
for(i = 1; i <= 3; i++)
{
   for(j = 1; j <= 3; j++)
   {
      printf("* ");
   }
   printf("\n");
}
```

Uses

- Pattern printing
- Matrix operations

8. Infinite Loops

A loop that **never terminates** is called an **infinite loop**.

Example

```
while(1)
{
   printf("Hello");
}
```

Reasons for Infinite Loops

- Missing increment/decrement
- Wrong condition

---

9. Loop Control Statements

9.1 break Statement

- Terminates the loop immediately

```
for(i = 1; i <= 10; i++)
{
   if(i == 5)
      break;
   printf("%d ", i);
}
```

9.2 continue Statement

- Skips current iteration and continues with next

```
for(i = 1; i <= 5; i++)
{
   if(i == 3)
      continue;
   printf("%d ", i);
}
```

---

10. Common Loop Programs

10.1 Print Numbers from 1 to N

```
for(i = 1; i <= n; i++)
   printf("%d ", i);
```

10.2 Sum of Natural Numbers

```
int sum = 0;
for(i = 1; i <= n; i++)
   sum += i;
```

10.3 Factorial of a Number

```
fact = 1;
for(i = 1; i <= n; i++)
   fact *= i;
```

---

11. Loop with Logical Conditions

Loops often use **relational and logical operators**.

Example

```
while(num > 0 && num < 100)
{
   printf("Valid number");
   break;
}
```

---

12. Common Errors in Loops

1. Infinite loops
2. Missing increment or decrement
3. Wrong condition
4. Extra semicolon after loop
5. Incorrect nesting

---

13. Advantages of Looping Statements

- Reduces code repetition
- Saves memory and time
- Improves readability
- Efficient execution

---

14. Limitations of Loops

- Complex logic may be hard to debug
- Infinite loops can crash programs
- Poorly designed loops reduce efficiency

---

15. Conclusion

Flow control loops are an essential part of C programming. The **while**, **do–while**, and **for** loops allow efficient execution of repetitive tasks. Proper understanding of loops helps in writing logical, optimized, and structured programs.