

Learning Objectives

By the end of this lesson, students will be able to:

1. Explain what a loop is.
2. Understand why loops are used in programming.
3. Identify repetitive tasks in real-life situations.
4. Compare code written with and without loops.
5. Understand the basic idea of repetition, a condition, and stopping a loop.

1. What Is a Loop?

A loop is a programming structure that repeats a block of code multiple times.

In real life, we often repeat the same task. For example:

- A teacher takes attendance for 50 students.
- A student practises a question 10 times.
- A university system sends notifications to many students.
- A program displays a message 5 times.

Instead of writing the same instructions repeatedly, we can use a loop.

Simple definition: A loop allows us to execute the same instructions repeatedly without writing them again and again.

2. Why Do We Need Loops?

Suppose we want to display a message five times.

Without a loop:

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Welcome to Ghazi University!" << endl;
    cout << "Welcome to Ghazi University!" << endl;
    cout << "Welcome to Ghazi University!" << endl;
    cout << "Welcome to Ghazi University!" << endl;
    cout << "Welcome to Ghazi University!" << endl;

    return 0;
}
```

Output:

```
Welcome to Ghazi University!  
Welcome to Ghazi University!  
Welcome to Ghazi University!  
Welcome to Ghazi University!  
Welcome to Ghazi University!
```

Problem: We have written the same instruction five times. What if we need to display it 100 times?

Writing 100 separate `cout` statements would be unnecessary and difficult to maintain.

We can solve this problem using a loop.

3. Our First Loop Program

We will use a `while` loop to display the same message five times.

```
#include <iostream>  
using namespace std;  
  
int main()  
{  
    int i = 1;  
  
    while (i <= 5)  
    {  
        cout << "Welcome to Ghazi University!" << endl;  
        i++;  
    }  
  
    return 0;  
}
```

Output:

```
Welcome to Ghazi University!  
Welcome to Ghazi University!  
Welcome to Ghazi University!  
Welcome to Ghazi University!  
Welcome to Ghazi University!
```

The output is the same, but the code is shorter and easier to manage.

4. Understand the Code Step by Step

Let's understand the program in simple language.

Step 1: Create a variable

```
int i = 1;
```

- `int` means the variable stores an integer.
- `i` is the variable name.
- `1` is its initial value.

Think of `i` as a counter that keeps track of how many times we have repeated the task.

Step 2: Check the condition

```
while (i <= 5)
```

The condition checks whether `i` is less than or equal to 5.

- If the condition is true, the loop body executes.
- If the condition is false, the loop stops.

Step 3: Execute the instruction

```
cout << "Welcome to Ghazi University!" << endl;
```

This statement displays the message.

Step 4: Update the counter

```
i++;
```

This increases `i` by 1.

For example:

```
i++;
```

is equivalent to:

```
i = i + 1;
```

Without this update, the condition would remain true indefinitely in this example, creating an infinite loop.

5. Trace Table: How Does the Loop Work?

A trace table helps us understand how a program executes step by step.

Iteration	Value of i	Condition i <= 5	Action
1	1	True	Display message
2	2	True	Display message
3	3	True	Display message
4	4	True	Display message
5	5	True	Display message
End check	6	False	Stop the loop

Notice that the loop checks the condition again after each update. When **i** becomes **6**, the condition is **false**, so the program exits the loop.

6. Understand Loops Through a Simple Analogy

Imagine placing five books on a table, one by one.

[Closed book on desk by blackboard](#)

1. Initialization

Start counting from book number 1.

[Books on the desk](#)

2. Condition

Ask: Have we placed five books or fewer?

[a diligent boy sits at a Desk and puts books on stacks.](#)

3. Repetition and update

Place another book and increase the count.

[Business Research Methods – Online Course – FutureLearn](#)

4. Stop

Stop when five books have been placed.

The same idea applies to programming: check, execute, update, and repeat until the condition becomes false.