# Function Overloading in C++

## Definition

Function overloading is a feature in C++ that allows multiple functions to have the same name but different parameter lists. The compiler determines which function to call based on the number and type of arguments passed.

## Key Rules

1. Functions must have the same name.
2. They must differ in the number or type of parameters.
3. Return type alone is NOT sufficient for overloading.
4. Function overloading works only in C++, not in C.

## Basic Example

```
int add(int a, int b) {
  return a + b;
}

double add(double a, double b) {
  return a + b;
}

int add(int a, int b, int c) {
  return a + b + c;
}
```

In this example, the compiler selects the correct add() function based on the arguments provided.

## How the Compiler Chooses

1. add(2, 3) → calls add(int, int)
2. add(2.5, 3.1) → calls add(double, double)
3. add(1, 2, 3) → calls add(int, int, int)

## Invalid Overloading (Error)

```
int test(int x);
double test(int x); // ERROR: return type only is different
```

The compiler cannot differentiate functions based only on return type.

## C vs C++

C does not support function overloading. In C, each function must have a unique name. C++ supports function overloading as a core language feature.

## Why Use Function Overloading?

1. Improves code readability.
2. Allows the same concept to use the same function name.
3. Commonly used in constructors, operators, and standard libraries.

## Summary

1. Same function name, different parameters.
2. Resolved at compile time.
3. Return type alone cannot overload a function.