C Programming - Functions & Beyond

By Himayoun

Previous Topics Covered

- What is C Programming?
- Variables & Data Types
- Operators & Arithmetic Expressions
- Conditional Statements (if-else, switch-case)
- Loops (for, while, do-while)

Topic for Today

- Functions in C

Topics We Are Covering in (maybe) next weeks (April)

- Arrays in C
- Strings in C
- Pointers in C
- Structures in C
- File Handling in C

Upcoming Topics in the Next Month (May)

- Dynamic Memory Allocation
- Preprocessors & Macros
- Advanced File Handling
- Data Structures in C (Stacks, Queues, Linked Lists)
- Recursion in C

What are Functions in C?

- A function is a **block of code** that performs a specific task.
- Helps in **code reusability** and **modularity**.
- Two types: **Library Functions** (printf, scanf) and **User-defined Functions**.

Defining and Calling Functions

- Functions are declared, defined, and called in C.

Syntax:

```
returnType functionName(parameters) {
    // Function body
    return value;
}
```

Example:

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

Function Parameters and Return Values

- Functions can **accept parameters** (inputs).
- Functions can **return values** using the return statement.
- Example:

```
int multiply(int x, int y) {
    return x * y;
}
```

Call by Value vs Call by Reference

- **Call by Value**: A copy of the actual argument is passed. Changes do not affect the original value.
- **Call by Reference**: The function receives a reference (address) to the actual argument. Changes affect the original value.
- Example:

```
void modify(int *p) {
    *p = 20; // Changes actual value
}
```

Example: Using Functions in C

```
#include <stdio.h>

// Function declaration
int add(int x, int y);

int main() {
    int a = 5, b = 10;
    int sum = add(a, b); // Function call
    printf("Sum: %d", sum);
    return 0;
}

// Function definition
int add(int x, int y) {
    return x + y;
}
```